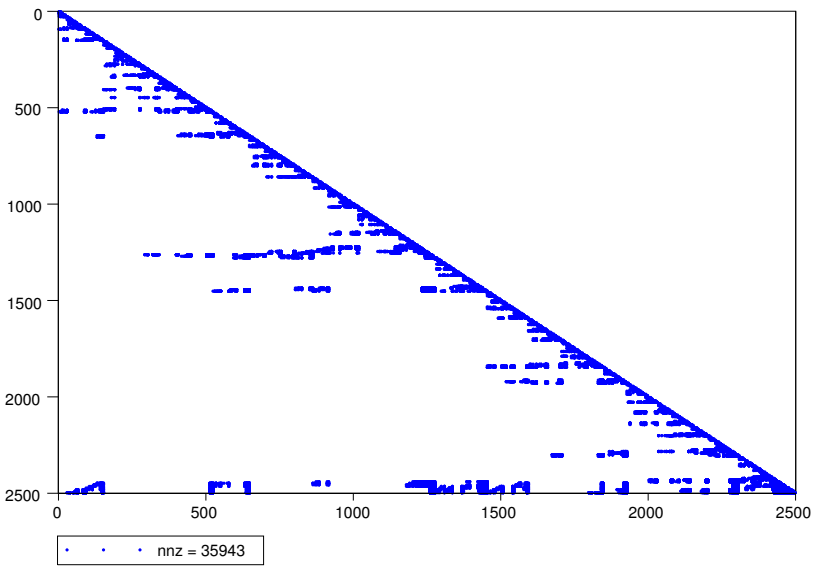
MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

Licence 3 Mathématiques et applications

MÉTHODES NUMÉRIQUES



Georges Koepfler 2023 - georges.koepfler@u-paris.fr

Table des matières

Avertissement : ces notes sont un support et complément du cours. Leur contenu n'est pas équivalent au cours enseigné pendant le semestre. Les examens et contrôles se basent sur le cours magistral, les travaux dirigés et pratiques.

Bibliographie.

Des références utiles pour suivre ce cours :

- M. Schatzman *Analyse numérique : une approche mathématique*, Dunod 2004.
- P.G. Ciarlet, *Introduction à l'analyse matricielle et à l'optimisation*, Masson 1990.

Quelques livres qui traitent de l'analyse numérique linéaire, et de ses applications :

- J. Demmel, *Applied Numerical Linear Analysis*, SIAM 1997 ;
- C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM 2000 ;
- P. Lascaux et J. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, 2 tomes, Masson 1988.
- G. H. Golub, C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1989.

Pour trouver des algorithmes en `C` et des références utiles, on pourra consulter

- W.H. Press, B.P. Flannery, S.A. Teukolsky et W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press 1988.

Chapitre 1

Introduction

Dans ce cours on va s'intéresser à des méthodes numériques pour calculer, ou approcher, la solution des problèmes tels la résolution de grands systèmes linéaires $Ax = b$, où A est une matrice réelle carrée ; la détermination de valeurs propres de la matrice A ; la résolution d'équations non linéaires $f(x) = 0$. On supposera en général que la solution existe, le but est de trouver des méthodes rapides et précises pour déterminer la ou les solutions.

Dans la résolution numérique d'un problème, il est essentiel de tenir compte des notions suivantes :

Instabilité du problème

Certains problèmes mathématiques sont instables : de petites perturbations des paramètres, dont dépend le problème, vont engendrer de grandes variations des solutions.
Des exemples sont la détermination des racines d'un polynôme, la résolution de l'équation de la chaleur inversée, certains systèmes dynamiques (modèle de Lorenz en météorologie).
Comme dans les applications les paramètres des modèles mathématiques viennent souvent d'expériences, donc avec une certaine imprécision, on utilise de préférence des modèles stables.
Soit $s = \mathcal{A}[e]$ un algorithme ou problème qui, en fonction du paramètre e en entrée, produit le résultat s. On s'intéresse à la stabilité de $\mathcal{A}$ par rapport à une perturbation δe de l'entrée. Pour cela on veut majorer l'erreur relative en sortie, grâce à l'erreur relative en entrée. Si l'on peut écrire

$$\frac{|\tilde{s} - s|}{|s|} = \frac{|\mathcal{A}(e + \delta e) - \mathcal{A}(e)|}{|\mathcal{A}(e)|} \leq c(\mathcal{A}) \, \frac{|\delta e|}{|e|} \, .$$

où la constante $c(\mathcal{A})$ est la meilleure borne possible pour obtenir cette inégalité. Alors on appelle $c(\mathcal{A})$ le *conditionnement du problème* $\mathcal{A}$.

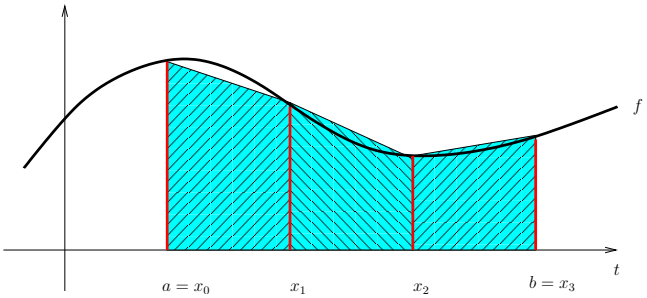
Erreurs d'approximation

Ce type d'erreur apparaît lorsque l'on remplace un problème continu par un problème discret. On est obligé à remplacer un objet mathématique, souvent défini grâce à des limites, par un nombre fini d'opérations numériques, on parle dans ce contexte aussi

d'*erreur de troncature*. Quand le pas de discrétisation tend vers zéro, c'est-à-dire le nombre d'opérations tend vers l'infini, la formulation discrète doit tendre vers la formulation continue du problème.
Les formules de quadrature pour calculer des intégrales sont des exemples de formulations discrètes. La méthode des trapèzes pour le calcul numérique de l'intégrale d'une fonction f sur $[a, b]$ s'écrit

$$\int_a^b f(t)dt \approx \sum_{i=0}^{N-1}(x_{i+1} - x_i)\frac{f(x_i) + f(x_{i+1})}{2}$$

Pour la partition $a = x_0 < x_1 < \cdots < x_N = b$, le pas de discrétisation est $h = \max_{0\leq i\leq N-1}(x_{i+1} - x_i)$.



Méthode du trapèze avec $N = 3$.

D'autres exemples sont les schémas aux différences finies pour la résolution d'équations différentielles et d'équations aux dérivées partielles.

Erreurs d'arrondi

Les calculs numériques sur ordinateur se font avec une précision finie. La mémoire disponible étant finie, on ne peut pas représenter les nombres réels qui ont un développement décimal infini. On peut représenter 1/3, mais pas son écriture décimale, on ne pourra représenter π que par un nombre fini de décimales.
Un représentation fréquente des nombres réels est celle en *virgule flottante normalisée* :

$$f = (-1)^s\ 0, d_{-1}d_{-2}\ldots d_{-r}\ \times\ b^j \quad \text{avec } m \leq j \leq M \text{ et } 0 \leq d_{-i} \leq b-1,$$

où s est le signe, $d_{-1}d_{-2}\cdots d_{-r}$ est la mantisse ou significande, b est la base et r est le nombre de chiffres significatifs. La *précision machine* pour un flottant de r chiffres significatifs est b^{1-r}, c'est la distance entre 1 et le nombre flottant immédiatement plus grand $1 + b^{1-r}$. Comme seulement un nombre fini de réels est représenté, le résultat des opérations arithmétiques de base $(+, -, \times, /)$ sur ces réels n'est pas nécessairement représentable, on est obligé d'arrondir.

L'arithmétique IEEE, utilisée sur les machines sous Unix et Linux, définit des nombres flottants en base $b = 2$: en simple précision de 32 bits (type `float` en C) et double précision de 64 bits (type `double` en C). En simple précision on utilise un bit pour s, 1 octet pour l'exposant (signé) j, *i.e.* $j \in \{-127, \ldots, +128\}$, et 23 bits pour la mantisse. Si l'on suppose que la représentation est normalisée, *i.e.* $d_{-1} \neq 0$, on gagne un bit significatif en utilisant la technique du bit caché et un flottant en simple précision s'écrit $f = (-1)^s\, 1, d_{-1}d_{-2}\ldots d_{-r} \times 2^j = (-1)^s\, 0, 1d_{-1}d_{-2}\ldots d_{-r} \times 2^{j+1}$.

La précision machine est alors 2^{-23}, c.-à-d. approximativement 10^{-7}, et les flottants normalisés positifs vont de 10^{-38} à 10^{+38}.

Les modules arithmétiques envoient aussi des messages qui indiquent la nature du résultat de l'opération : si ce n'est pas un nombre (`NaN` = *NotANumber*) résultat de "$\infty - \infty$" ou "$\sqrt{-1}$" ou "0/0"... ; si le résultat est un nombre trop grand pour être représenté, `+/-INF`.

Dans `Octave` aussi, les nombres réels sont représentés en virgule flottante en base 2, avec $r = 53$. La précision machine est stockée dans la constante `eps` et vaut $2^{-52} \sim 2.22\,10^{-16}$.

Complexité des calculs

Pour la plupart des applications, on veut obtenir un résultat en un temps raisonnable, il est donc important de pouvoir estimer le temps que l'algorithme va utiliser pour résoudre le problème. Pour cela, on compte le nombre d'opérations flottantes (angl. *flops*) en fonction de la taille du problème. Souvent il suffit d'avoir un ordre de grandeur, si N est la taille du problème, p.ex. le nombre d'inconnues, on peut avoir des algorithmes en $O(N)$, $O(N \ln N)$, $O(N^2)$, $O(N^3)$, $O(N!)$...

Ordre d'une méthode itérative

Mis à part le nombre d'opérations total à exécuter, il peut être intéressant de comparer des vitesses de convergence.

Soit $(x^{(n)})_{n\in\mathbb{N}}$ une suite convergeant vers x^*. On dit que la convergence est *d'ordre* $\lambda \geq 1$, s'il existe une constante $C \in \mathbb{R}_+^*$ telle que, pour n suffisamment grand

$$\frac{\|x^{(n+1)} - x^*\|}{\|x^{(n)} - x^*\|^\lambda} \leq C \, .$$

Si $\lambda = 1$, il faut $C \in]0, 1[$ pour avoir convergence et on a alors *convergence linéaire*.

Si $\lambda = 2$ on a *convergence quadratique*.

La quantité $(-\log_{10} \|x^{(n)} - x^*\|)$ mesure le nombre de décimales exactes derrière la virgule, la quantité $(-\log_{10} \left(\|x^{(n)} - x^*\| / \|x^*\| \right))$ indique le nombre de chiffres exact.

Pour une méthode d'ordre 1, on a

$$(-\log_{10} \|x^{(n+1)} - x^*\|) \geq (-\log_{10} \|x^{(n)} - x^*\|) - \log_{10}(C)$$

on gagne à chaque itération $\log_{10}(1/C)$ décimales.

Pour une méthode d'ordre $\lambda > 1$, on a

$$(-\log_{10} \|x^{(n+1)} - x^*\|) \geq \lambda\,(-\log_{10} \|x^{(n)} - x^*\|) - \log_{10}(C)$$

alors x_{n+1} a λ fois plus de décimales exactes que x_n.

Ainsi une méthode quadratique, $\lambda = 2$, double avec chaque itération le nombre chiffres exacts.

Exemples

Les quatre exemples qui suivent illustrent les divers problèmes posés par les notions précédentes.

Exemple 1 : Détermination des racines d'un polynôme

Pour illustrer l'instabilité de la résolution d'une équation polynomiale par rapport aux coefficients du polynôme, on va présenter l'exemple de Wilkinson (1963) . On définit un polynôme d'ordre 20 comme suit :

$$\begin{aligned} w(x) &= \prod_{i=1}^{20}(x-i) \\ &= x^{20} - 210\,x^{19} + 20615\,x^{18} - 1256850\,x^{17} + 53327946\,x^{16} - 1.672\cdot 10^{9}\,x^{15} \\ &\quad +4.017\cdot 10^{10}\,x^{14} - 7.561\cdot 10^{11}\,x^{13} + 1.131\cdot 10^{13}\,x^{12} - 1.356\cdot 10^{14}\,x^{11} \\ &\quad +1.308\cdot 10^{15}\,x^{10} - 1.014\cdot 10^{16}\,x^{9} + 6.303\cdot 10^{16}\,x^{8} - 3.113\cdot 10^{17}\,x^{7} \\ &\quad +1.207\cdot 10^{18}\,x^{6} - 3.600\cdot 10^{18}\,x^{5} + 8.038\cdot 10^{18}\,x^{4} - 1.287\cdot 10^{19}\,x^{3} \\ &\quad +1.380\cdot 10^{19}\,x^{2} - 8.753\cdot 10^{18}\,x + 20! \end{aligned}$$

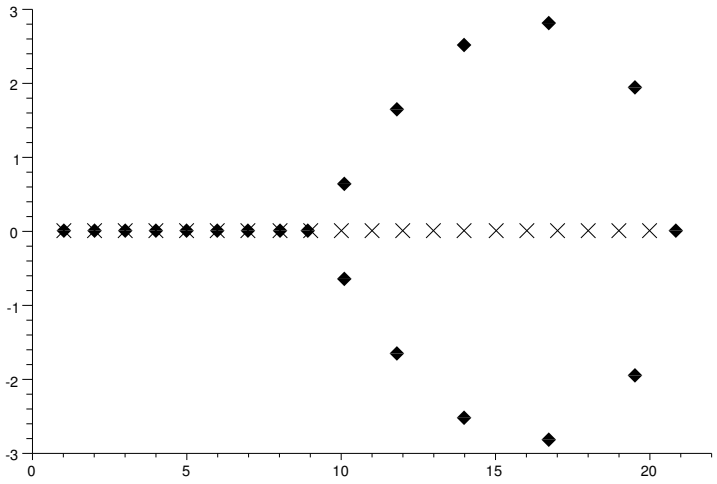
On considère ensuite le polynôme perturbé suivant $\tilde{w}(x) = w(x) - 2^{-23}\,x^{19}$. En faisant des calculs numériques en haute précision, on obtient les racines suivantes pour $\tilde{w}$:

1, 00000 0000	10, 09526 6145 + 0, 64350 0904 i
2, 00000 0000	10, 09526 6145 − 0, 64350 0904 i
3, 00000 0000	11, 79363 3881 + 1, 65232 9728 i
4, 00000 0000	11, 79363 3881 − 1, 65232 9728 i
4, 99999 9928	13, 99235 8137 + 2, 51883 0070 i
6, 00000 6944	13, 99235 8137 − 2, 51883 0070 i
6, 99969 7234	16, 73073 7466 + 2, 81262 4894 i
8, 00726 7603	16, 73073 7466 − 2, 81262 4894 i
8, 91725 0249	19, 50243 9400 + 1, 94033 0347 i
20, 84690 8101	19, 50243 9400 − 1, 94033 0347 i

Cet exemple célèbre montre comment une petite perturbation d'un coefficient ($2^{-23} \approx 10^{-7}$) transforme un polynôme ayant 20 racines réelles en un polynôme avec 10 racines réelles et 10 racines complexes conjuguées de partie imaginaire non négligeable. L'ensemble des solutions d'une équation polynomiale dépend de façon continue mais instable des coefficients du polynôme.

Au polynôme $p(x) = \sum_{i=0}^{d} a_i\,x^i$ on associe sa *matrice compagne* $M_p \in \mathcal{M}(d,d)$ définie par

$$M_p = \begin{pmatrix} -\frac{a_{d-1}}{a_d} & -\frac{a_{d-2}}{a_d} & & -\frac{a_2}{a_d} & -\frac{a_1}{a_d} & -\frac{a_0}{a_d} \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & & 0 & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{pmatrix}$$



Les zéros de w ($\times$) et de $\tilde{w}$ ($\blacklozenge$) dans le plan complexe.

Les valeurs propres de M_p sont les racines de p. La détermination de toutes les valeurs propres d'une matrice est un problème difficile qui sera abordé plus loin.

Exemple 2 : Évaluation d'un polynôme proche d'une racine multiple

Soit $p(x) = \sum_{i=0}^{d} a_i\, x^i$ un polynôme à coefficients réels avec $a_d \in \mathbb{R}^*$.
Pour calculer la valeur de p en x, en utilisant cette représentation, on doit effectuer de l'ordre de $2d$ opérations élémentaires ($+$ et $\times$) et $d-1$ appels à la fonction puissance.

Si l'on connaît toutes les racines du polynôme, on peut factoriser et écrire $p(x) = a_d \prod_{i=1}^{d}(x - r_i)$.
Il suffit alors de $2d$ opérations élémentaires pour évaluer p en x. Comme l'on connaît rarement une factorisation du polynôme, on utilise de façon générale l'*algorithme de* Horner pour calculer $p(x)$.

Algorithme 1.1 (Horner)
Évaluation d'un polynôme en x à partir de ses coefficients $a_0, a_1, \ldots, a_d$.

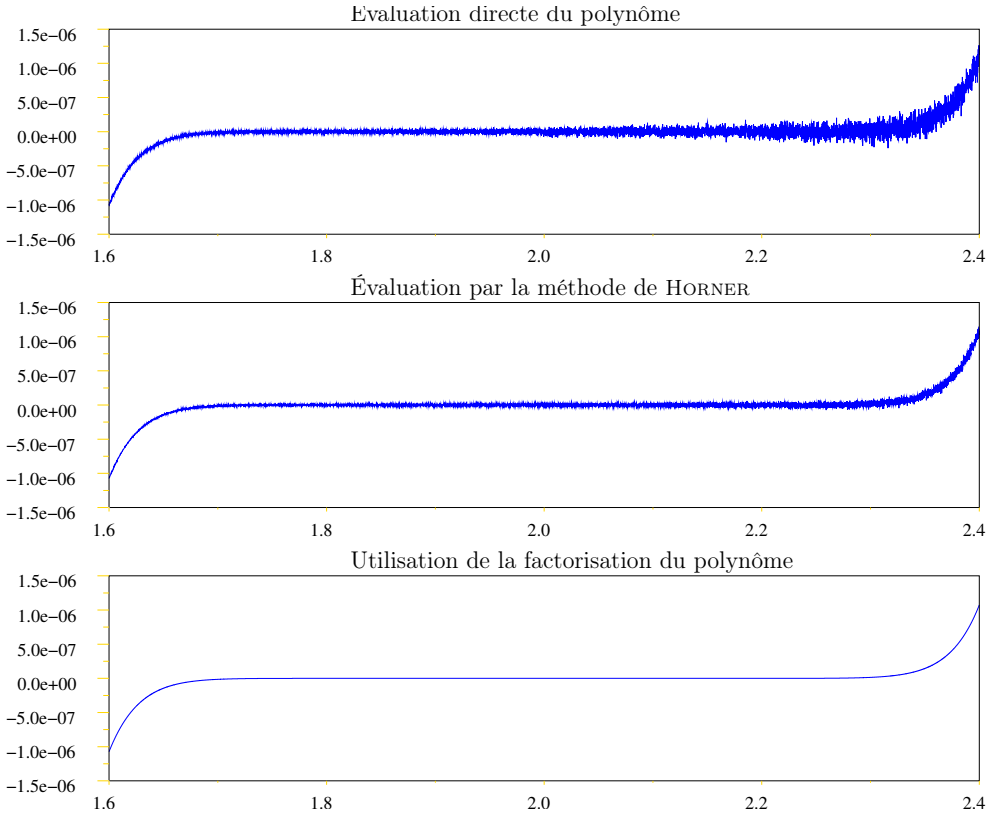
$y = a_d$
pour *i=d-1* ***à*** *0*
  $y = x \times y + a_i$

Cet algorithme permet d'évaluer $p(x)$ à partir des coefficients a_i en $2d$ opérations élémentaires. On se propose de calculer et représenter le polynôme suivant :

$$\begin{aligned} p(x) &= x^{15} - 30x^{14} + 420x^{13} - 3640x^{12} + 21840x^{11} - 96096x^{10} \\ &\quad +320320x^9 - 823680x^8 + 1647360x^7 - 2562560x^6 + 3075072x^5 \\ &\quad -2795520x^4 + 1863680x^3 - 860160x^2 + 245760x - 32768 \\ &= (x-2)^{15}\ . \end{aligned}$$

Les résultats sont représentés à la figure suivante, on a évalué $p(x)$ pour $x \in [1.6, 2.4]$ et avec un pas de 10^{-4}. En fonction de la méthode d'évaluation choisie, on aura des erreurs d'arrondi plus ou moins importantes comme l'on peut constater sur les graphes.

Les oscillations qui apparaissent dans le premier graphe rendent difficile la détection de la racine $x = 2$ par une méthode comme celle de la dichotomie par exemple.
Pour calculer de façon rapide et précise les valeurs d'un polynôme il vaut donc mieux utiliser sa forme factorisée, or pour cela il faut trouver tous les zéros de l'équation polynomiale $p(x) = 0$, ce qui est instable comme l'on a vu avec l'exemple de Wilkinson.



Exemple 3 : Calcul de la trajectoire de la fusée ARIANE 5

Le 4 juin 1996, le premier vol de la fusée ARIANE 5 s'est terminé par l'explosion de l'engin à peine 50s après le décollage. Dans le rapport ESA/CNES [1], établi suite à cet incident, on peut lire que les causes de l'échec sont dues à un signal d'*overflow* mal interprété par l'ordinateur de bord.

En effet, les deux systèmes de référence inertiels ont arrêté de fonctionner à cause d'une erreur lors d'une conversion d'un nombre flottants en 64 bit vers un entier signé en 16 bits. Le nombre à convertir ayant une valeur trop grande pour être représenté en 16 bits, l'opération s'est terminé par un signal d'*overflow*. Le premier système de référence inertiel a arrêté de fonctionner et le second à pris le relais, et la même erreur c'est produite. L'ordinateur de bord a cette fois pris en compte le signal d'*overflow*, mais en l'interprétant

1. *ARIANE 5 : Flight 501 Failure*, rapport sous la direction de J.L. Lions, juillet 1996.

comme étant une donnée. Il y a eu un changement de trajectoire qui a mené la fusée vers une forte déviation, ceci a causé la désintégration du lanceur.

Le rapport des enquêteurs constate que le programme en cause provenait d'un code transféré sans changement du système ARIANE 4 vers ARIANE 5, sans tenir compte du fait que sur le nouveau lanceur les paramètres physiques prenaient des valeurs plus importantes.

Conclusion de ce feu d'artifice fort coûteux : il est important de connaître des bornes pour les valeurs d'entrée d'un programme afin de réserver assez de mémoire et de protéger les logiciels contre d'éventuels résultats erronés.

Exemple 4 : Multiplication de matrices

Soient A, B et C des matrices rectangulaires, de dimensions respectives (n, m), (m, p) et (p, q) . Combien d'opérations sont nécessaires pour calculer ABC ?

Comme $ABC = (AB)C = A(BC)$, on a deux possibilités d'évaluation.

Le calcul de $(AB)C$ nécessite de l'ordre de $O(2np(m+q))$ opérations, tandis que le calcul de $A(BC)$ se fait en $O(2mq(n+p))$ opérations. Si l'on prend par exemple $n = p = 10$ et $m = q = 100$ on trouve $4 \cdot 10^4$ et $4 \cdot 10^5$.

La multiplication des matrices rectangulaires étant distributive, on obtient deux fois le même résultat, mais le nombre d'opérations nécessaires peut varier de façon importante.

Si $n = m = p$, alors A et B sont des matrices carrées et le coût de leur multiplication est $O(n^3)$. Il existe des algorithmes permettant de réduire le nombre d'opérations. Présentons d'abord une version récursive de la multiplication matricielle habituelle.

Algorithme 1.2
Calcul par blocs du produit de deux matrices carrées d'ordre $n = 2^r$: $C = AB$
$C = MULT\_BLOC(A, B)$

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ et } B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad // \text{ où les } A_{ij} \text{ et } B_{ij} \text{ sont } [n/2, n/2] //$$

$$C_{11} = A_{11} * B_{11} + A_{12} * B_{21}$$
$$C_{12} = A_{11} * B_{12} + A_{12} * B_{22}$$
$$C_{21} = A_{21} * B_{11} + A_{22} * B_{21}$$
$$C_{22} = A_{21} * B_{12} + A_{22} * B_{22}$$

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Comme les matrices sont carrées d'ordre n, avec $n = 2^r$, on peut appliquer l'algorithme de façon récursive. À chaque itération on effectue ainsi 8 produits de matrices de dimensions moitié et le nombre d'opérations élémentaires est donc $O(8^r) = O(n^3)$.

La méthode de Strassen (1969) est basée sur une réécriture de la multiplication par blocs de deux matrices. Comme l'on peut voir, à chaque itération on effectue seulement 7 produits de matrices de dimensions moitié et le nombre d'opérations élémentaires est donc $O(7^r) = O(n^{\log_2 7}) = O(n^{2,8073})$.

Algorithme 1.3 (Strassen)
Calcul récursif du produit de deux matrices carrées d'ordre $n = 2^r$: $C = AB$
$C = STRASSEN(A, B, n)$

si $n = 1$ ***alors***
$C = A \times B$ // *cas scalaire* //
sinon
$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ *et* $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$ // *où les* A_{ij} *et* B_{ij} *sont* $[n/2, n/2]$ //
$Q_1 = STRASSEN(A_{11} + A_{22}, B_{11} + B_{22}, n/2)$
$Q_2 = STRASSEN(A_{21} + A_{22}, B_{11}, n/2)$
$Q_3 = STRASSEN(A_{11}, B_{12} - B_{22}, n/2)$
$Q_4 = STRASSEN(A_{22}, -B_{11} + B_{21}, n/2)$
$Q_5 = STRASSEN(A_{11} + A_{12}, B_{22}, n/2)$
$Q_6 = STRASSEN(-A_{11} + A_{21}, B_{11} + B_{12}, n/2)$
$Q_7 = STRASSEN(A_{12} - A_{22}, B_{21} + B_{22}, n/2)$
$C_{11} = Q_1 + Q_4 - Q_5 + Q_7$
$C_{12} = Q_3 + Q_5$
$C_{21} = Q_2 + Q_4$
$C_{22} = Q_1 - Q_2 + Q_3 + Q_6$

$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

En pratique, il faut que n soit assez grand pour obtenir un gain satisfaisant car sinon les nombreuses additions diminuent les performances. Pour n suffisamment grand, on constate que cette méthode, d'apparence compliquée, est plus rapide que la méthode classique.

Des méthodes simples de multiplication par blocs sont par utilisées dans les librairies BLAS (angl. *Basic Linear Algebra Subroutines*).

La vitesse de calcul et la mémoire centrale des processeurs ne suffisent pas pour obtenir des programmes rapides. Si l'on doit multiplier deux grandes matrices dont la taille nécessite une sauvegarde dans une mémoire d'accès lent, il est alors possible que la plus grande partie du temps est utilisée pour transférer des données. Les librairies tels que BLAS implémentent des multiplications de matrices par blocs en tenant compte de l'architecture du processeur. Les blocs transférés de la mémoire lente sont d'une taille adaptée à la mémoire d'accès rapide du processeur, on minimise le nombre de transferts lents et on passe proportionnellement plus de temps sur les calculs.

Les librairies LAPACK, CLAPACK et ScaLAPACK (`http://www.netlib.org`) utilisent ces techniques. Des logiciels tels que `Octave` [2] ou `Matlab`©, utilisent des opérations par blocs pour accroître leurs performances.

2. logiciel disponible à `https://www.gnu.org/software/octave/`

Chapitre 2

Algèbre linéaire appliquée

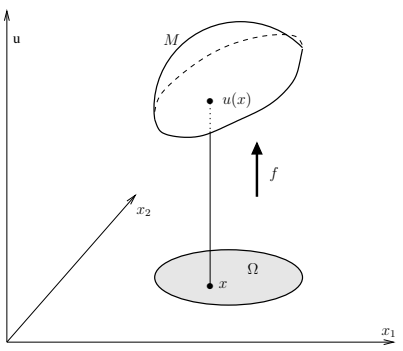
Dans la première section de ce chapitre, on présente un exemple de modélisation d'un problème d'équilibre. On montre que la résolution numérique se ramène à la résolution d'un système linéaire $Ax = b$ avec un grand nombre d'inconnues x, donc une matrice A de grandes dimensions.
Ceci motive la suite du chapitre qui propose d'abord une introduction à l'analyse matricielle. On s'intéresse ensuite aux méthodes de résolution de l'équation $Ax = b$ en grande dimension. On va étudier des méthodes directes et des méthodes itératives. On termine avec quelques méthodes de calcul de valeurs propres de la matrice A.

2.1 Origine de grands systèmes linéaires

Les problèmes qui se ramènent finalement à la résolution d'un système linéaire sont nombreux. On peut y aboutir par interpolation, par des problèmes d'optimisation, par la résolution d'équations différentielles ou aux dérivées partielles On présente dans la suite un exemple de résolution d'une équation aux dérivées partielles en deux dimensions.

2.1.1 Modélisation

On considère une membrane élastique qui s'identifie à une région plane $\Omega \subset \mathbb{R}^2$ si on n'applique aucune force. Si on applique une force f, normale à $\mathbb{R}^2$, le membrane se déforme et prend une position d'équilibre M.
On note $u(x) \in M$ la position d'équilibre du point $x = (x_1, x_2) \in \Omega$, donc $M = u(\Omega)$. On suppose qu'il n' a que des petites déformations longitudinales.



Le travail de la tension est proportionnel à la variation d'aire de la membrane. Si $\tau \in \mathbb{R}_+^*$, cette variation peut s'écrire

$$\tau\big(aire(M) - aire(\Omega)\big) = \int_\Omega \tau\Big(\sqrt{1+|\nabla u|^2} - 1\Big)\, dx \ .$$

Comme $|u_{x_1}|$ et $|u_{x_2}|$ sont supposés petits, on a $\sqrt{1+|\nabla u|^2} \approx 1 + |\nabla u|^2/2$.
Le travail de la force f étant donné par le déplacement des points $x \in \Omega$, on obtient finalement l'expression suivante de l'énergie potentielle de la membrane à l'équilibre :

$$E(u) = \int_\Omega \left(-\frac{1}{2}\tau|\nabla u|^2 + fu\right)\, dx\,.$$

Le calcul des variations permet de montrer que cette énergie est minimale pour la solution de l'*équation de Poisson* dans un ouvert Ω :

$$\tau\Delta u + f = 0 \qquad \text{dans } \Omega\,.$$

Des contraintes sur le bord de la membrane M induisent des contraintes sur u :

- si le bord de la membrane est fixé, $u(x) = b(x)$ pour $x \in \partial\Omega$ et on obtient le *problème de Dirichlet*
$$\begin{cases} -\Delta u = \dfrac{1}{\tau} f & \text{sur } \Omega \\ u = b & \text{sur } \partial\Omega \end{cases}$$
- si le bord de la membrane n'est pas fixé, on peut vérifier que l'on a des contraintes sur le gradient de u et que u est solution du *problème de Neumann* avec conditions au bord homogènes
$$\begin{cases} -\Delta u = \dfrac{1}{\tau} f & \text{sur } \Omega \\ \dfrac{\mathrm{d}u}{\mathrm{d}n} = 0 & \text{sur } \partial\Omega \end{cases}$$

2.1.2 Discrétisation

Pour simplifier, on va supposer que $\tau = 1$, $\Omega =]0,1[^2$ et que $u = 0$ sur $\partial\Omega$, *i.e.* un problème de Dirichlet homogène, c'est-à-dire que le bord de la membrane reste fixe.

$$\begin{cases} -\Delta u = f & \text{sur } \Omega =]0,1[^2 \\ u = 0 & \text{sur } \partial\Omega = \{(0,y),(1,y),(x,0),(x,1) \ / \ 0 \le x,y \le 1\} \end{cases}$$

L'inconnue est la fonction u définie pour $(x,y) \in \Omega$. Le domaine de définition Ω de u est discrétisé par la *grille uniforme*

$$G_h = \{(x_m, y_n) \,/\, x_m = mh,\ y_n = nh,\ 0 \le m,n \le N+1\}$$

où $h = \frac{1}{N+1}$ et $N+1$ est le nombre d'intervalles de longueur h de la partition :

$$[0,h] \cup [h,2h] \cup \cdots \cup [(N-1)h, Nh] \cup [Nh, (N+1)h] = [0,1]$$

On dit que h est le *pas de discrétisation spatiale.*

La fonction u qui est définie pour les variables continues (x, y) prend la valeur $u_{m,n} = u(mh, nh)$ au point $(x_m, y_n) = (mh, nh)$ de la grille G_h.
On note de même $f_{m,n} = f(mh, nh)$.

Pour discrétiser l'équation aux dérivées partielles, on va remplacer les dérivées partielles par des *différences finies*. On a les approximations classiques suivantes :

- $\dfrac{\partial u}{\partial x}(x_m, y_n) \simeq \dfrac{u_{m+1,n} - u_{m,n}}{h}$, différence finie progressive ;
- $\dfrac{\partial u}{\partial x}(x_m, y_n) \simeq \dfrac{u_{m-1,n} - u_{m,n}}{h}$, différence finie rétrograde ;
- $\dfrac{\partial u}{\partial x}(x_m, y_n) \simeq \dfrac{u_{m+1,n} - u_{m-1,n}}{2h}$, différence finie centrée ;
- $\dfrac{\partial^2 u}{\partial x^2}(x_m, y_n) \simeq \dfrac{u_{m+1,n} - 2u_{m,n} + u_{m-1,n}}{h^2}$, différence finie centrée d'ordre deux.

On obtient ainsi l'approximation du Laplacien par un schéma centré :

$$\begin{aligned}\Delta u(x_m, y_n) &= \frac{\partial^2 u}{\partial x^2}(x_m, y_n) + \frac{\partial^2 u}{\partial y^2}(x_m, y_n) \\ &\simeq \frac{1}{h^2}\left(u_{m+1,n} + u_{m-1,n} + u_{m,n+1} + u_{m,n-1} - 4u_{m,n}\right)\end{aligned}$$

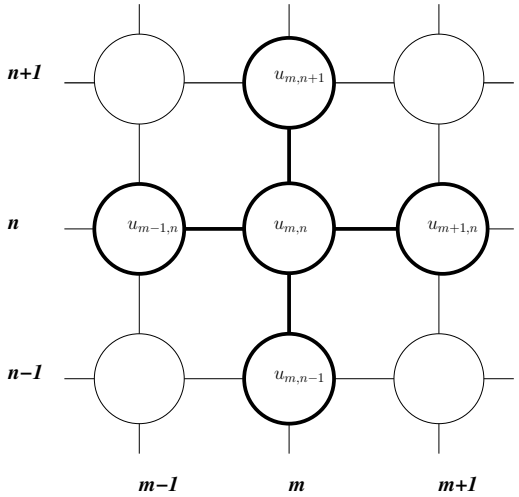


Schéma à 5 points pour le calcul du Laplacien.

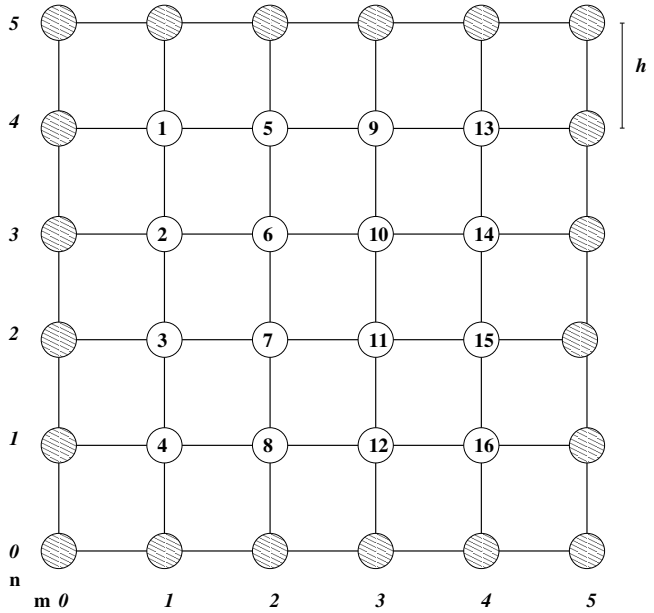
Le problème discret étant une approximation du problème continu, on note $v_{m,n}$, $0 \leq m, n \leq N+1$, les variables discrètes afin de distinguer les deux problèmes et leur solutions. Quand h est petit, on veut avoir $v_{m,n} \simeq u_{m,n} = u(mh, nh)$. En étudiant les schémas aux différences finies, on peut obtenir des résultats qui assurent la convergence de $(v_{m,n})_{(m,n)\in\mathbb{Z}^2}$ vers $(u_{m,n})_{(m,n)\in\mathbb{Z}^2}$ quand h tend vers 0.

Comme $u = 0$ sur $\partial\Omega$ on a $v_{m,0} = v_{m,N+1} = v_{0,n} = v_{N+1,n} = 0$ pour $0 \leq m, n \leq N+1$, il faut donc déterminer les N^2 inconnues $v_{m,n}$ pour $1 \leq m, n \leq N$ grâce aux équations

$$\frac{1}{h^2}\left(-v_{m+1,n} - v_{m-1,n} - v_{m,n+1} - v_{m,n-1} + 4v_{m,n}\right) = f_{m,n}\,, \text{ pour } 1 \leq m, n \leq N \quad (2.1)$$

2.1.3 Résolution numérique

On peut réécrire les N^2 équations sous la forme d'un système linéaire $Ax = h^2 b$ en numérotant $v_{m,n}$ et $f_{m,n}$. Par exemple de haut en bas et colonne par colonne, ce système est adapté au fonctionnement de `Octave`, mais d'autres numérotations sont possibles.
De façon précise, on a $x_k = v_{m,n}$ et $b_k = f_{m,n}$
où $k = (N + 1 - n) + N(m - 1) = Nm - n + 1$, pour $1 \leq m, n \leq N$.



Grille G_h pour $N = 4$ et numérotation, colonne par colonne, des 16 inconnues.
Les cercles gris indiquent les valeurs aux bord, fixées à 0.

Les équations précédentes en $v_{m,n}$ et $f_{m,n}$, s'écrivent alors

$$\frac{1}{h^2}\left(-x_{k+N} - x_{k-N} - x_{k-1} - x_{k+1} + 4x_k\right) = b_k\,.$$

D'où $a_{k,k} = 4$ et $a_{k,k-N} = a_{k,k+N} = a_{k,k-1} = a_{k,k+1} = -1$, les autres éléments de la k^{e} ligne de A sont nuls.
Si l'on note I_N la matrice identité d'ordre N et D la matrice tridiagonale d'ordre N définie par

$$D = \begin{pmatrix} 4 & -1 & 0 & & \cdots & & 0 \\ -1 & 4 & -1 & 0 & \cdots & & 0 \\ 0 & -1 & 4 & -1 & & & \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ & & & & & & 0 \\ 0 & & & & -1 & 4 & -1 \\ 0 & & & & 0 & -1 & 4 \end{pmatrix}$$

alors A est la matrice d'ordre N^2 définie par blocs

$$A = \begin{pmatrix} D & -I_N & O & & \cdots & & O \\ -I_N & D & -I_N & O & \cdots & & O \\ O & -I_N & D & -I_N & & & \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ & & & & & & O \\ O & & & & -I_N & D & -I_N \\ O & & & & O & -I_N & D \end{pmatrix}$$

Pour $N = 4$, on a la matrice d'ordre 16

$$A = \left(\begin{array}{cccc|cccc|cccc|cccc} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 \end{array}\right)$$

On est donc amené à résoudre le système linéaire $Ax = h^2 b$ à N^2 inconnues.

Si l'on prend $h = 10^{-2}$ alors $N \simeq 10^2$ et A est d'ordre 10^4, si l'on prend $h = 10^{-3}$, on a $N \simeq 10^3$ et A est d'ordre 10^6.

En utilisant la méthode de Cramer et le calcul récursif d'un déterminant, on a un nombre d'opérations élémentaires en $O(N^2!)$, ce qui est infaisable dès que $N = 10$. Il faut donc développer des méthodes efficaces et précises pour résoudre les grands systèmes linéaires.

Exemple : Dans ce cas simple, il est possible de trouver la solution exacte de l'équation aux dérivées partielles sous forme d'une série de Fourier, pour $(x_1, x_2) \in [0, 1]^2$:

$$u(x_1, x_2) = \frac{16f}{\pi^4} \sum_{m,n \in 2\mathbb{N}+1} \frac{1}{mn(m^2+n^2)} \sin(\pi m x_1) \sin(\pi n x_2)\,.$$

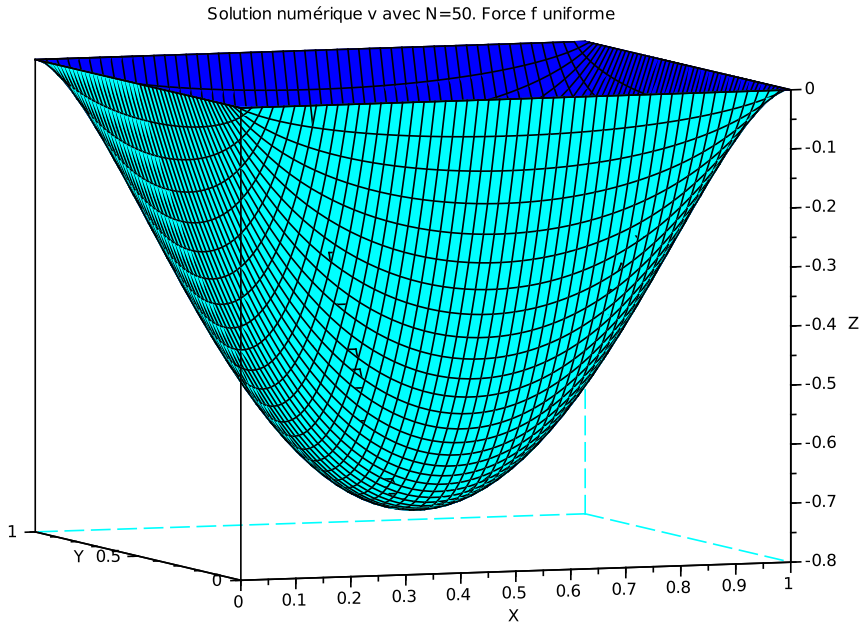
On peut donc comparer la solution numérique v et la solution analytique u.

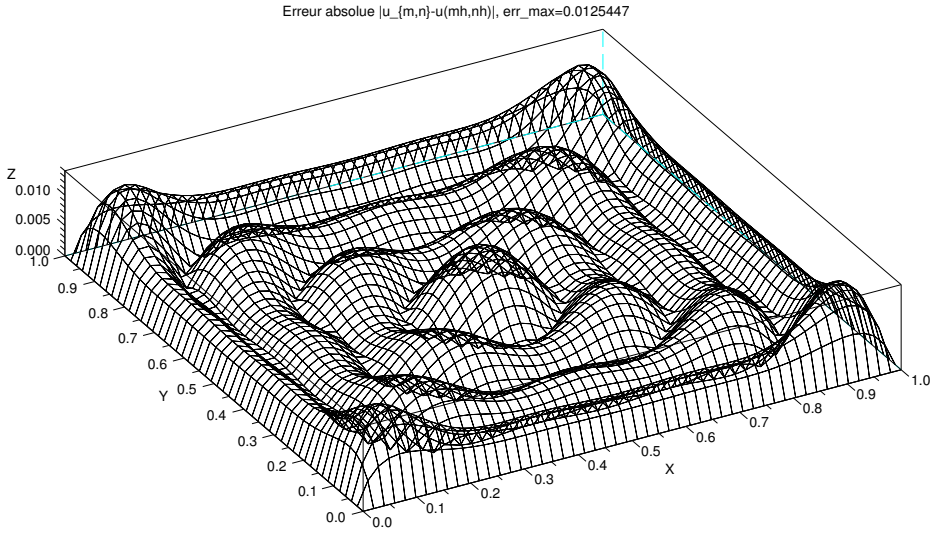
On va calculer la solution numérique avec $f(x, y) = -10$ sur $]0, 1[^2$.

On a pris $N = 50$, *i.e.* $h \simeq 0,02$, la matrice A est alors de taille 2500×2500. Pour la résolution du système linéaire $Ax = h^2 b$, on a utilisé la décomposition de Cholesky de `Octave` pour des *matrices creuses*. En effet, on peut montrer que A est symétrique définie positive et que seulement 12300 coefficients, sur 2500^2, sont non nuls, donc à peine 2%.

On peut aussi utiliser les méthodes itératives (Jacobi, Gauss-Seidel) ou des méthodes adaptées aux matrices tridiagonales par blocs.

Sur la figure suivante, on a représenté la solution du problème de Dirichlet avec $f(x,y) = -10$ sur $]0,1[^2$. La seconde figure représente l'erreur $|v_{m,n} - u(mh,nh)|$, on constate que l'erreur maximale en valeur absolue est de l'ordre de $0,013$.





2.2 Algèbre linéaire et analyse matricielle

2.2.1 Rappels. Définitions

Soit V un espace vectoriel de dimension finie sur $\mathbb{R}$ ou sur $\mathbb{C}$. Un vecteur v de V est identifié au vecteur (ou matrice) colonne de ses coordonnées $v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$, le vecteur transposé est un vecteur (ou matrice) ligne $v^t = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix}$.

On note $A = (a_{ij})$ une matrice et $a_{ij} = (A)_{ij}$ est l'élément de la i-ième ligne et j-ième colonne pour $1 \leq i \leq m$ et $1 \leq j \leq n$. Les dimensions de A sont alors $m \times n$ ou $[m, n]$, en `Octave`, la commande `size(A)` retourne le résultat `[m,n]`.
Les majuscules seront en général réservées aux matrices et les minuscules aux éléments des matrices. Un scalaire est une matrice de taille $[m, n] = [1, 1]$, $A = (a_{11})$.
Attention : on note la matrice transposée de A par $A^t = (a_{ji})$ (norme ISO 80000-2 :2019), ce qui se rapproche de l'opération de transposition en `Octave A'`.

Si A et B sont deux matrices, alors AB désigne le produit matriciel, défini seulement si les dimensions de A et B sont compatibles. En `Octave` on écrit `A*B` .

Si $u, v \in V$, e.v. sur $\mathbb{R}$, le produit $u^t\, v = \sum_{i=1}^{n} u_i v_i =< u, v >$, est un nombre réel (matrice $[1, 1]$), c'est le produit scalaire euclidien. Par contre $u\, v^t = (u_i v_j)_{1 \leq i,j \leq n}$ est une matrice $n \times n$. Si la multiplication est définie $(AB)^t = B^t A^t$.

Si V est un e.v. sur $\mathbb{C}$, on considère la matrice adjointe de A, donnée par $A^* = (\overline{a_{ji}})$ et le produit hermitien de deux vecteurs $< u, v >= v^*\, u = \overline{u^*\, v} = \sum_{i=1}^{n} u_i \overline{v_i}$. Si la multiplication est définie $(AB)^* = B^* A^*$.

Une matrice carrée A d'ordre n est inversible s'il existe une matrice, notée A^{-1}, vérifiant $AA^{-1} = A^{-1}A = I_n$, où I_n est la matrice identité. On a $(A^t)^{-1} = (A^{-1})^t$ et $(A^*)^{-1} = (A^{-1})^*$. Si B est aussi d'ordre n et inversible, alors $(AB)^{-1} = B^{-1}A^{-1}$.

Soit A une matrice carrée réelle, alors

$$\begin{array}{ll} A \text{ est } \textit{symétrique} & \text{si } A^t = A\,; \\ A \text{ est } \textit{orthogonale} & \text{si } AA^t = A^t\, A = I_n. \end{array}$$

Soit A une matrice carrée complexe, alors

$$\begin{array}{ll} A \text{ est } \textit{hermitienne} & \text{si } A^* = A\,; \\ A \text{ est } \textit{unitaire} & \text{si } AA^* = A^*\, A = I_n\,; \\ A \text{ est } \textit{normale} & \text{si } AA^* = A^*\, A. \end{array}$$

On dit qu'une matrice A réelle symétrique est *définie positive* si pour tout $w \in \mathbb{R}^n$, $w \neq 0$, $w^t Aw > 0$; la matrice A est *semi-définie positive* si pour tout $w \in \mathbb{R}^n$, $w^t Aw \geq 0$;.

Le *spectre* de la matricé carrée A est le sous-ensemble de $\mathbb{C}$:

$$\sigma(A) = \text{spec}(A) = \{\lambda \in \mathbb{C} \,/\, \det(A - \lambda I_n) = 0\} = \{\lambda_1, \ldots, \lambda_n\}.$$

Le *rayon spectral* de A est $\rho(A) = \max\{|\lambda| \,/\, \lambda \in \mathrm{spec}(A)\} = \max_{1\leq i\leq n}\{|\lambda_1|, \ldots, |\lambda_n|\}$.

Théorème 2.2.1 (Réduction de matrices)

1. *Soit A une matrice carrée quelconque, il existe alors une matrice unitaire U telle que U^*AU soit triangulaire (décomposition de* Schur*).*
2. *Si de plus A est une matrice normale, il existe alors une matrice unitaire U telle que U^*AU soit diagonale.*
3. *Cas réel : si A est une matrice symétrique, il existe alors une matrice orthogonale O telle que O^tAO soit diagonale.*

Pour simplifier un calcul, une démonstration ou présenter un algorithme, on considère souvent les décompositions par blocs de matrices.

Définition 2.2.1 (Matrices blocs)
Soit A une matrice de taille $[m, n]$, si $m = m_1 + m_2 + \cdots + m_p$ et $n = n_1 + n_2 + \cdots + n_q$, alors on peut représenter A par blocs

$$A = \begin{pmatrix} A_{11} & \ldots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \ldots & A_{pq} \end{pmatrix}$$

où les blocs, ou sous-matrices, A_{ij} ont comme dimensions $[m_i, n_j]$. Il y a pq blocs.

Calculs avec les matrices blocs :

1. Soient A et B des matrices de même dimensions et avec les blocs respectifs A_{ij} et B_{ij} de même dimensions. Alors on obtient l'*addition par blocs* de A et B :

$$A + B = \begin{pmatrix} A_{11} + B_{11} & \ldots & A_{1q} + B_{1q} \\ \vdots & & \vdots \\ A_{p1} + B_{p1} & \ldots & A_{pq} + B_{pq} \end{pmatrix} .$$

2. Soient A et B des matrices tel que AB est défini et

$$A = \begin{pmatrix} A_{11} & \ldots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \ldots & A_{pq} \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} B_{11} & \ldots & B_{1s} \\ \vdots & & \vdots \\ B_{r1} & \ldots & B_{rs} \end{pmatrix} ,$$

si $q = r$ et si les produits $A_{ik}B_{kj}$ $(i = 1, \ldots, i\,;\ j = 1, \ldots, s\,;\ k = 1, \ldots, q)$ sont définis, alors la *multiplication par blocs* de A et B s'écrit :

$$AB = \begin{pmatrix} \sum_{k=1}^{q} A_{1k}B_{k1} & \ldots & \sum_{k=1}^{q} A_{1k}B_{ks} \\ \vdots & & \vdots \\ \sum_{k=1}^{q} A_{pk}B_{k1} & \ldots & \sum_{k=1}^{q} A_{pk}B_{ks} \end{pmatrix} .$$

3. Une matrice carré A de taille $n \times n$ est *diagonale par blocs* si les blocs A_{ii} sont carrés, $m_i = n_i$, et si tous les autres blocs sont des matrices nulles, $A_{ij} = O$ pour $i \neq j$.

$$A = \begin{pmatrix} A_{11} & O & O & \dots & O \\ O & A_{22} & O & \dots & O \\ O & O & A_{33} & & \vdots \\ \vdots & \vdots & & \ddots & \\ O & O & \dots & & A_{pp} \end{pmatrix}$$

La matrice A est la somme directe des $A_{11}, A_{22}, \dots, A_{pp}$ et on a

$$\det(A) = \prod_{i=1}^{p} \det(A_{ii}) \,, \quad \operatorname{trace}(A) = \sum_{i=1}^{p} \operatorname{trace}(A_{ii}) \,.$$

Si A est inversible, on a l'*inverse par blocs*

$$A^{-1} = \begin{pmatrix} A_{11} & O & O & \dots & O \\ O & A_{22} & O & \dots & O \\ O & O & A_{33} & & \vdots \\ \vdots & \vdots & & \ddots & \\ O & O & \dots & & A_{pp} \end{pmatrix}^{-1} = \begin{pmatrix} A_{11}^{-1} & O & O & \dots & O \\ O & A_{22}^{-1} & O & \dots & O \\ O & O & A_{33}^{-1} & & \vdots \\ \vdots & \vdots & & \ddots & \\ O & O & \dots & & A_{pp}^{-1} \end{pmatrix}$$

Exemple : Cas particulier $p = q = r = 2$, $s = 1$, alors, si les blocs sont compatibles, on a

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathcal{M}(n,n) \,, x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^n \text{ et } Ax = \begin{pmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{pmatrix} \in \mathbb{R}^n \,.$$

Si le bloc A_{21} est nul, *i.e.* la matrice A est triangulaire supérieure par blocs,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ O & A_{22} \end{pmatrix} \,, \quad \text{alors} \quad \operatorname{spec}(A) = \operatorname{spec}(A_{11}) \cup \operatorname{spec}(A_{22}) \,.$$

2.2.2 Normes vectorielles. Normes matricielles

Définition 2.2.2 (Norme vectorielle)
Une norme vectorielle sur l'espace vectoriel V est une application de V vers $\mathbb{R}_+$ vérifiant, pour tous $u, v \in V$, $\alpha \in \mathbb{R}$ (ou $\mathbb{C}$) :

$$\begin{aligned} &\|v\| = 0 \Leftrightarrow v = 0 && \textit{(séparation)} \\ &\|\alpha v\| = |\alpha| \, \|v\| && \textit{(absolument homogène)} \\ &\|u + v\| \leq \|u\| + \|v\| && \textit{(sous-additive)} \end{aligned}$$

Définition 2.2.3 (Norme matricielle)
Une norme matricielle sur l'e.v. des matrices carrées d'ordre n, $\mathcal{M}_n$, est une application de $\mathcal{M}_n$ vers $\mathbb{R}_+$ vérifiant, pour tous $A, B \in \mathcal{M}_n$, $\alpha \in \mathbb{R}$ (ou $\mathbb{C}$) :

$$\begin{aligned} &\|A\| = 0 \Leftrightarrow A = 0 \\ &\|\alpha A\| = |\alpha| \, \|A\| \\ &\|A + B\| \leq \|A\| + \|B\| \\ &\|AB\| \leq \|A\| \, \|B\| && \textit{(sous-multiplicative)} \end{aligned}$$

Définition 2.2.4 (Norme matricielle subordonnée)
Soit $\|.\|$ une norme vectorielle, on appelle norme matricielle subordonnée (à cette norme vectorielle) l'application de $\mathcal{M}_n$ vers $\mathbb{R}_+$ définie par

$$|||A||| = \sup_{x\in V, x\neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{x\in V, \|x\|=1} \|Ax\| = \sup_{x\in V, \|x\|\leq 1} \|Ax\| .$$

Proposition 2.2.1
Soit $|||.|||$ une norme matricielle subordonnée à la norme vectorielle $\|.\|$

1. *Les trois définitions sont équivalentes et il existe $u \in V$ avec $\|u\| = 1$ tel que $|||A||| = \|Au\|$.*
   Comme le sup *est atteint, on peut le remplacer par un* max.
2. *Pour tous $A \in \mathcal{M}_n$, $v \in V$: $\|Av\| \leq |||A||| \, \|v\|$.*
3. *On a $|||I_n||| = 1$.*
4. *Une norme matricielle subordonnée est une norme matricielle.*

Remarque : dans la suite on va abandonner la notation $|||.|||$ et utiliser la notation classique $\|.\|$ pour les normes matricielles subordonnées. Le contexte permet de savoir si on a affaire à une norme de vecteur ou de matrice.

Proposition 2.2.2 (Exemples fondamentaux)
Soit $v \in \mathbb{R}^n$ et A une matrice carrée réelle, pour les normes vectorielles usuelles on a les normes matricielles subordonnées suivantes :

$$\|v\|_\infty = \max_{1\leq i\leq n} |v_i| \qquad \text{et} \quad \|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| \quad \text{«maximum sur les lignes»}$$

$$\|v\|_1 = \sum_{1\leq i\leq n} |v_i| \qquad \text{et} \quad \|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| \quad \text{«maximum sur les colonnes»}$$

$$\|v\|_2 = \left(\sum_{1\leq i\leq n} v_i^2 \right)^{1/2} \qquad \text{et} \quad \|A\|_2 = \sqrt{\rho(A^t A)} \qquad \text{où } \rho(A^t A) \text{ est le rayon spectral de la matrice symétrique semi-définie positive } A^t A .$$

Proposition 2.2.3 (Propriétés)
Soit A une matrice carrée réelle, alors

1. $\|A\|_1 = \|A^t\|_\infty$
2. $\|A\|_2 = \|A^t\|_2$
3. *Si U est orthogonale, $\|UA\|_2 = \|AU\|_2 = \|A\|_2$*
4. *Si A est normale, alors $\|A\|_2 = \rho(A)$*
   Cas particuliers : si A est symétrique, $\|A\|_2 = \rho(A)$,
   si A est orthogonale, $\|A\|_2 = 1$

Remarque : On peut avoir des normes sur l'e.v. des matrices qui ne sont pas des normes matricielles car non compatibles avec la multiplication matricielle. De même il existe des normes matricielles non subordonnées, comme le montre le résultat suivant.

PROPOSITION 2.2.4 (NORME DE FROBENIUS)
La norme de FROBENIUS *définie, pour toute matrice $A \in \mathcal{M}_n$, par*

$$\|A\|_F = \left(\sum_{1 \leq i,j \leq n} |a_{ij}|^2 \right)^{\frac{1}{2}} = (trace(A^t\, A))^{\frac{1}{2}} \,,$$

est une norme matricielle, non subordonnée qui vérifie :

1. *Si U est une matrice orthogonale, alors $\|UA\|_F = \|AU\|_F = \|A\|_F$.*
2. *On a l'encadrement $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\, \|A\|_2$.*
3. *Pour tout $x \in V$, $\|Ax\|_2 \leq \|A\|_F \|x\|_2$.*

THÉORÈME 2.2.2
Soit $A \in \mathcal{M}_n$, alors

1. *Pour toute norme matricielle $\|.\|$ on a $\rho(A) \leq \|A\|$.*
2. *Pour tout $\varepsilon \in \mathbb{R}_+^*$, il existe une norme matricielle subordonnée $\|.\|_\varepsilon$ telle que $\|A\|_\varepsilon \leq \rho(A) + \varepsilon$.*

PROPOSITION 2.2.5 (CS)
Soit $A \in \mathcal{M}_n$ et $\|.\|$ une norme matricielle, on a

1. *Si $\|A\| < 1$, alors $\lim_{k \to +\infty} A^k = O$.*
2. *Si $I_n - A$ est une matrice singulière, alors $\|A\| \geq 1$.*
3. *Si $\|A\| < 1$, alors $I_n - A$ est inversible et*

$$(I_n - A)^{-1} = \sum_{k=0}^{+\infty} A^k \quad \text{avec} \quad \|(I_n - A)^{-1}\| \leq \frac{1}{1 - \|A\|} \,.$$

THÉORÈME 2.2.3 (CNS)
Soit $A \in \mathcal{M}_n$, alors

1. $\rho(A) < 1 \iff \lim_{k \to +\infty} A^k = O \iff \lim_{k \to +\infty} A^k v = 0$, *pour tout* $v \in V$
2. $\rho(A) < 1 \iff \sum_{k=0}^{+\infty} A^k = (I_n - A)^{-1}$.

Le rayon spectral est donc l'information déterminante pour étudier la convergence de suites et séries de matrices. La norme d'une matrice permet souvent de conclure, mais l'on a seulement une condition suffisante.

THÉORÈME 2.2.4 (GERSHGORIN, 1931)
Soit $A \in \mathcal{M}_n$, pour $i \in \{1, \ldots, n\}$ on définit les disques de Gershgorin

$$D_i = \left\{ z \in \mathbb{C}, |a_{ii} - z| \leq \sum_{j \neq i} |a_{ij}| \right\} = D(a_{ii}, R_i) \subset \mathbb{C}, \quad \text{où } R_i = \sum_{j \neq i} |a_{ij}| \,.$$

Alors

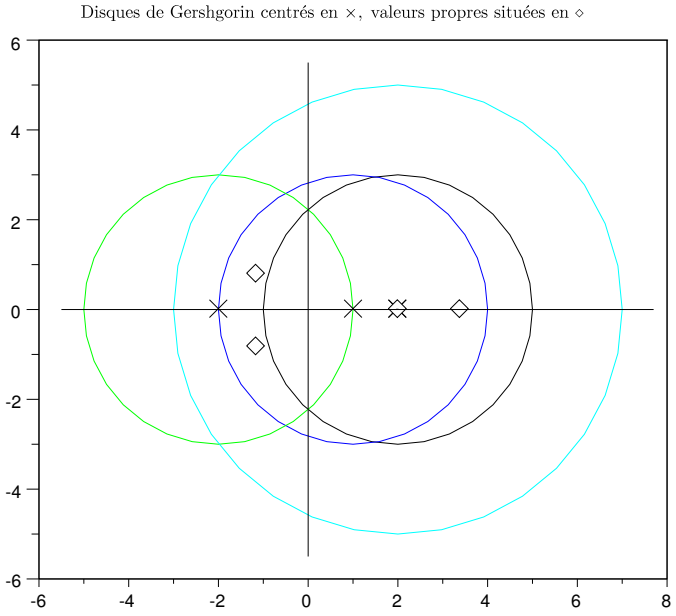
$$\text{spec}(A) \subset \bigcup_{1 \leq i \leq n} D(a_{ii}, R_i) \subset \mathbb{C} \,.$$

THÉORÈME 2.2.5
Soit $A \in \mathcal{M}_n$ et $\|.\|$ une norme matricielle, alors

$$\lim_{k\to+\infty} \|A^k\|^{1/k} = \rho(A)\,.$$

Exemple : On considère la matrice $A = \begin{pmatrix} 2 & 1 & 2 & 0 \\ 1 & 1 & -1 & -1 \\ -1 & 0 & -2 & -2 \\ -2 & 1 & -2 & 2 \end{pmatrix}$

dont le spectre est $\text{spec}(A) = \{-1.19 \pm i\,0.81;\ 3.38\,;\ 2\}$ et $\rho(A) = 3.38$.



2.2.3 Stabilité de l'inversion matricielle et conditionnement

On suppose dans toute la suite que A est une matrice carrée inversible et on s'intéresse à la stabilité de la solution de l'équation $Ax = b$ pour des données perturbées.
Pour cela on considère le problème perturbé $(A + \delta A)\tilde{x} = b + \delta b\,,$
où $\delta A \in \mathcal{M}_n$ et $\delta b \in \mathbb{R}^n$ sont les perturbations, ou erreurs sur les données. On s'intéresse en particulier à l'erreur relative commise sur la solution $\dfrac{\|\tilde{x} - x\|}{\|x\|}$. Dans la suite on note $\delta x = \tilde{x} - x$.

DÉFINITION 2.2.5 (CONDITIONNEMENT)
Le conditionnement de la matrice A, pour le problème d'inversion et pour la norme $\|.\|$ est le nombre positif $c(A) = \|A^{-1}\|\,\|A\|\,.$

On note $c_i(A) = \|A^{-1}\|_i\,\|A\|_i$ pour $i \in \{1, 2, \infty\}$.

PROPOSITION 2.2.6 (PROPRIÉTÉS)
Soit A une matrice carrée inversible et $\|.\|$ une norme matricielle subordonnée, alors

1. $c(I_n) = 1$; $c(A) \geq 1$; $c(\alpha A) = c(A)$ *pour* $\alpha \in \mathbb{R}^*$;
2. $c(AB) \leq c(A)c(B)$, *pour* $B \in \mathcal{M}_n$ *inversible* ;
3. *Si* A *est normale, alors* $c_2(A) = \dfrac{\max_{1\leq i\leq n} |\lambda_i|}{\min_{1\leq i\leq n} |\lambda_i|}$.
4. *Si* U *est une matrice orthogonale, alors* $c_2(U) = 1$ *et* $c_2(U^tAU) = c_2(A)$.

THÉORÈME 2.2.6
Soit A *une matrice inversible, alors*

$$\min \left\{ \frac{\|\delta A\|_2}{\|A\|_2} \Big/ A + \delta A \text{ singulière} \right\} = \frac{1}{\|A^{-1}\|_2 \|A\|_2} = \frac{1}{c_2(A)} .$$

La distance relative de A *à la matrice singulière la plus proche est l'inverse du conditionnement de* A *en* $\|.\|_2$.

Le conditionnement a une influence directe sur la solution du système linéaire $Ax = b$ comme le montre l'exemple suivant.

Exemple : On considère la matrice $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$ avec $\det(A) = +1$.

La matrice inverse est aussi à coefficients entiers et vaut $A^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{pmatrix}$.

Posons maintenant

$$\delta A = \begin{pmatrix} -0.002553422381 & 0.004227714089 & -0.001061727529 & 0.000629732580 \\ 0.004227714089 & -0.006999847167 & 0.001757907531 & -0.001042651353 \\ -0.001061727529 & 0.001757907531 & -0.000441472337 & 0.000261846383 \\ 0.000629732580 & -0.001042651353 & 0.000261846383 & -0.000155306511 \end{pmatrix}$$

et $B = A + \delta A$, les coefficients de la matrice A sont perturbés au plus à l'ordre 10^{-3}. Mais la matrice B est (presque) singulière, en effet $\det(B) = 3.797 \cdot 10^{-14}$.
La matrice A est donc proche d'une matrice singulière, elle est mal conditionnée et en effet $c_2(A) = 2984.1$.

Examinons maintenant la solution du système linéaire $Ax = b$ et celle du système perturbé $B\tilde{x} = \tilde{b}$.

Si l'on résout le système $Ax = b$ avec $b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$, on obtient $x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Pour $\delta b = \begin{pmatrix} +0.5 \\ -0.5 \\ +0.5 \\ -0.5 \end{pmatrix}$, on a $\tilde{b} = b + \delta b = \begin{pmatrix} 32.5 \\ 22.5 \\ 33.5 \\ 30.5 \end{pmatrix}$ et on obtient comme solution $\tilde{x} = \begin{pmatrix} 42 \\ -67 \\ 18.5 \\ -9.5 \end{pmatrix}$.

Mêmes des petites perturbations de A et b engendrent de grandes variations de la solution.

Les résultats suivants précisent le rôle joué par le conditionnement d'une matrice.

THÉORÈME 2.2.7
Soit $\|.\|$ une norme matricielle subordonnée, soient $A, \delta A \in \mathcal{M}_n$ et $b, \delta b, x, \tilde{x} \in \mathbb{R}^n$, avec A inversible et b non nul, vérifiant

$$Ax = b \quad \text{et} \quad (A + \delta A)\tilde{x} = b + \delta b \; .$$

On pose $\delta x = \tilde{x} - x$, alors, $$\frac{\|\delta x\|}{\|x + \delta x\|} \leq c(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\|\|x + \delta x\|} \right) \; .$$

Si l'on a $\|\delta A\| < \dfrac{1}{\|A^{-1}\|}$ (cf. *théorème 2.2.6*) *, alors*

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{c(A)}{1 - c(A)\frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right) \; .$$

En particulier, si $\delta A = O$: $$\frac{\|\delta x\|}{\|x\|} \leq c(A) \frac{\|\delta b\|}{\|b\|}.$$

PROPOSITION 2.2.7
Soit $\|.\|$ une norme matricielle subordonnée, soient $A \in \mathcal{M}_n$ inversible et $b, x, \tilde{x} \in \mathbb{R}^n$, vérifiant $Ax = b$ et $A\tilde{x} - b = r \neq 0$ où r est le vecteur résidu.

Si $b \neq 0$ et $\delta x = \tilde{x} - x$, alors $$\frac{\|\delta x\|}{\|x\|} \leq c(A) \frac{\|r\|}{\|b\|} \; .$$

Exemple : Pour $\varepsilon \in]0, 1[$, on pose $A = \begin{pmatrix} 1 & -1 \\ 1 & -1 + \varepsilon \end{pmatrix}$.

Comme $\det(A) = \varepsilon \neq 0$, le système $Ax = b$ a une solution unique pour tout vecteur b.
On montre que $c(A) = O(\varepsilon^{-1})$, donc plus ε est petit, plus A est mal conditionnée.

Pour $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ la solution exacte du système est $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Supposons que la solution numérique obtenue est $\tilde{x} = x + \varepsilon^{-\frac{1}{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, c'est-à-dire que $\dfrac{\|\delta x\|}{\|x\|} = O(\varepsilon^{-\frac{1}{2}})$.

Le vecteur résidu $r = \varepsilon^{-\frac{1}{2}} A \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \varepsilon^{+\frac{1}{2}} \end{pmatrix}$ et $\dfrac{\|r\|}{\|b\|} = O(\varepsilon^{+\frac{1}{2}})$.

Donc pour une matrice mal conditionnée, une mauvaise solution numérique $\tilde{x}$, avec une erreur relative importante, peut tout de même donner lieu à un résidu faible. Ce qui est cohérent avec l'inégalité de la proposition précédente.

2.3 Résolution de systèmes linéaires par des méthodes directes

Une méthode directe de résolution de $Ax = b$ donnerait, en absence d'erreurs d'arrondi, le résultat exact en un nombre fini d'opérations.

Parmi les nombreuses méthodes directes existantes, on va présenter quelques unes parmi les plus classiques. En fonction des propriétés et de la structure de la matrice A, on peut souvent trouver une méthode adaptée.

2.3.1 Systèmes triangulaires

Rappelons brièvement les algorithmes de résolution de systèmes triangulaires inférieurs et supérieurs qui sont particulièrement peu coûteux en nombre d'opérations.

Soit $A \in \mathcal{M}_n$ une matrice triangulaire inférieure, *i.e.* tous les coefficients au-dessus de la diagonale sont nuls, $a_{ij} = 0$ pour $1 \leq i < j \leq n$, le système linéaire s'écrit

$$\left\{\begin{array}{rcccccccc} a_{11}x_1 & & & & & & & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & & & & & = & b_2 \\ \vdots & & \vdots & & & & & & \vdots \\ a_{n-1,1}x_1 & + & a_{n-1,2}x_2 & + \cdots + & a_{n-1,n-1}x_n & & & = & b_{n-1} \\ a_{n1}x_1 & + & a_{n2}x_2 & + \cdots + & a_{n,n-1}x_n & + & a_{nn}x_n & = & b_n \end{array}\right.$$

Si $\det(A) = \prod_{i=1}^{n} a_{ii} \neq 0$, alors la solution de $Ax = b$ est donnée par

Algorithme 2.1
Algorithme progressif de résolution d'un système triangulaire inférieur.

$$x_1 = \frac{b_1}{a_{11}}$$

pour $i = 2$ ***à*** n

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right)$$

On vérifie que cet algorithme est en $O(n^2)$ opérations.

Soit $A \in \mathcal{M}_n$ une matrice triangulaire supérieure, $a_{ij} = 0$ pour $1 \leq j < i \leq n$, *i.e.* tous les coefficients au-dessous de la diagonale sont nuls. Si tous les éléments de la diagonale sont non nuls, on obtient l'algorithme, de complexité $O(n^2)$, suivant.

Algorithme 2.2
Algorithme rétrograde de résolution d'un système triangulaire supérieur.

$$x_n = \frac{b_n}{a_{nn}}$$

pour $i = n - 1$ ***à*** 1

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right)$$

2.3.2 Factorisation LU

Parmi les innombrables factorisations possibles d'une matrice A, celle qui permet de décomposer A sous la forme d'un produit LU, où L est une matrice triangulaire inférieure (*angl. Lower*) et U une matrice triangulaire supérieure (*angl. Upper*), est très utilisée pour la résolution de systèmes linéaires et pour le calcul du déterminant et a été introduite en 1938 par Tadeusz Banachiewicz.

Définition 2.3.1 (Matrice de permutation)
Une matrice de permutation P est une matrice identité où l'on a permuté les colonnes. On écrit $I_n = [e_1\, e_2 \cdots e_n]$, où e_j est le j^e vecteur colonne de la base canonique et l'on construit P en échangeant les colonnes de I_n.

Proposition 2.3.1 (Propriétés)
- *Une matrice de permutation est une matrice orthogonale $P^{-1} = P^t$.*
- *On a $\det(P) = (-1)^s$ où $s \in \mathbb{N}$ est la signature de la permutation.*
- *Le produit de deux matrices de permutation est encore une matrice de permutation.*
- *Soit $A \in \mathcal{M}_n$, alors PA est la matrice A avec une permutation des lignes.*
- *Soit $A \in \mathcal{M}_n$, alors AP est la matrice A avec une permutation des colonnes.*

Exemple : Soit $P = [e_3\, e_1\, e_2] = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ et $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$.

Alors $Px = \begin{pmatrix} x_2 \\ x_3 \\ x_1 \end{pmatrix}$, $x^t P = (x_3\, x_1\, x_2)$, $P^t = P^{-1} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ et $\det(P) = (-1)^2 = 1$.

Théorème 2.3.1
Soit $A \in \mathcal{M}_n$ réelle et inversible, on a équivalence des affirmations

1. *Il existe une unique décomposition $A = LU$, où L est une matrice triangulaire inférieure, avec $l_{ii} = 1$ pour $1 \leq i \leq n$, et U est une matrice triangulaire supérieure avec $\prod_{i=1}^{n} u_{ii} \neq 0$.*
2. *Toutes les sous-matrices principales (ou diagonales) de A, $M_m = \left(a_{ij}\right)_{1 \leq i,j \leq m}$ pour $1 \leq m \leq n$, sont inversibles.*

La matrice P de l'exemple précédent est inversible mais ne vérifie pas le point 2 du théorème précédent. D'où l'utilité du résultat suivant.

Théorème 2.3.2
Soit A une matrice inversible, alors il existe des matrices de permutations P_1 et P_2, une matrice L triangulaire inférieure avec des 1 sur la diagonale et une matrice U triangulaire supérieure inversible, telles que

$$P_1\, A\, P_2 = L\, U\,.$$

En fait une seule matrice de permutation est nécessaire, de plus

1. *on peut choisir $P_2 = I$ et P_1 tel que a_{11} soit l'élément de valeur absolue maximale dans la première colonne, c'est l'algorithme d'élimination de* Gauss *avec pivot partiel.*
2. *on peut choisir P_1 et P_2 de façon à ce que a_{11} soit l'élément de valeur absolue maximale dans toute la matrice, c'est l'algorithme d'élimination de* Gauss *avec pivot total.*

Algorithme 2.3
Factorisation LU avec pivot et destruction de A

$U = O_n$, $L = I_n$
pour $i = 1$ **à** $n-1$
  Appliquer les permutations à A, L et U pour que $a_{ii} \neq 0$.
  pour $j = i+1$ **à** n
    $l_{ji} = a_{ji}/a_{ii}$
  pour $j = i$ **à** n
    $u_{ij} = a_{ij}$
  pour $j = i+1$ **à** n
    pour $k = i+1$ **à** n
      $a_{jk} = a_{jk} - l_{ji} \cdot u_{ik}$
$u_{nn} = a_{nn}$

On vérifie que le coût de cet algorithme de factorisation est $O(\frac{2}{3}n^3)$. Les permutations éventuelles ne nécessitent pas d'opérations arithmétiques.

Résolution du système linéaire $Ax = b$

Algorithme 2.4
Résoudre $Ax = b$ par élimination de Gauss *avec pivot partiel*

Factoriser A en $PA = LU$;
Par permutation : $LUx = Pb$;
Résoudre un système triangulaire inférieur : $Ly = Pb$;
Résoudre un système triangulaire supérieur : $Ux = y$;
Par permutation : $x = A^{-1}b = U^{-1}L^{-1}Pb$.

La résolution du système $Ax = b$ revient donc à la factorisation LU de coût $O(\frac{2}{3}n^3)$ et à la résolution de deux systèmes triangulaires de coût $O(n^2)$. Le coût total de l'élimination de Gauss pour résoudre un système linéaire est donc $O(\frac{2}{3}n^3)$.

Remarques :

1. Pour calculer le déterminant d'une matrice A on utilise $\det(A) = \det(LU) = \prod_{i=1}^{n} u_{ii}$.
   Le coût est $O(\frac{2}{3}n^3)$, ce qui est donc beaucoup moins que la méthode par développement récursif du déterminant, méthode infaisable en pratique car en $O(n!)$.

2. Pour résoudre $Ax = b$ on ne calcule pas la matrice inverse A^{-1} !
   En effet, la détermination de A^{-1} nécessite la résolution de n systèmes linéaires $Ax_j = e_j$, avec e_j, $1 \leq j \leq n$, le j^{e} vecteur colonne de la base canonique et $A^{-1} = [x_1 \cdots x_n]$.

3. La factorisation LU est une méthode générique. Dans les applications, on rencontre souvent des matrices ayant une structure particulière : des matrices symétriques, bande, tridiagonales, tridiagonales par blocs, creuses ...
   Avec des méthodes adaptées, on peut souvent améliorer les performances des algorithmes de factorisation, que ce soit au niveau de l'espace mémoire, du nombre d'opérations ou du transfert de données.
   La plupart des librairies scientifiques (*e.g.* LAPACK, CLAPACK) ou logiciels (*e.g.* `Octave`, `Matlab`) proposent des algorithmes et représentations pour les types de matrices les plus fréquents (*e.g.* matrices creuses).

2.3.3 Factorisation de Cholesky

Dans beaucoup d'applications la matrice A est symétrique définie positive. Dans ce cas, on économise de l'espace mémoire en ne stockant que $n(n+1)/2$ termes. De plus il existe un algorithme de factorisation en $O(\frac{1}{3}n^3)$. Le théorème suivant regroupe quelques résultats utiles.

THÉORÈME 2.3.3

1. *Si $A = (a_{ij})_{1\leq i,j\leq n}$ est une matrice symétrique définie positive, alors toute sous-matrice $M_m = (a_{ij})_{1\leq i,j\leq m}$ est symétrique définie positive.*
2. *Soit B une matrice inversible, alors A est symétrique définie positive si et seulement si B^tAB est symétrique définie positive.*
3. *Une matrice A est symétrique définie positive si et seulement si $A^t = A$ et toutes ses valeurs propres sont strictement positives.*
4. *Si A est symétrique définie positive,*
   alors $a_{ii} > 0$ pour $1 \leq i \leq n$ et $\max\limits_{1\leq i,j\leq n} |a_{ij}| = \max\limits_{1\leq i\leq n} a_{ii} > 0$.
5. *Une matrice A est symétrique définie positive si et seulement si il existe une matrice L triangulaire inférieure telle que $A = L\,L^t$. La matrice L est unique et on a pour tout $1 \leq i \leq n$, $l_{ii} > 0$.*
   C'est la factorisation de CHOLESKY *(1910) de la matrice A.*

ALGORITHME 2.5 (CHOLESKY)
Factorisation de CHOLESKY *d'une matrice symétrique définie positive.*

pour $j = 1$ ***à*** n

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{\frac{1}{2}}$$

pour $i = j+1$ ***à*** n

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$$

On montre que la complexité de cet algorithme est $O(\frac{1}{3}n^3)$. On n'a pas besoin de pivot pour garantir la stabilité de l'algorithme. Un version modifiée qui donne $A = \tilde{L}D\tilde{L}^t$ permet de s'affranchir de l'extraction de racines carrées (D est diagonale et $\tilde{L}$ triangulaire inférieure avec des 1 sur la diagonale).

Pour résoudre $Ax = LL^t x = b$ on doit résoudre deux systèmes triangulaires en plus de la factorisation, le coût total de la résolution est donc de $O(\frac{1}{3}n^3)$.

La décomposition de Cholesky, adaptée aux matrices creuses, a été utilisée pour la résolution du système linéaire dans l'exemple de la page 13.

2.4 Résolution de systèmes linéaires par des méthodes itératives

Les méthodes directes présentées jusqu'ici permettent de calculer la solution d'un système linéaire en un nombre fini de pas (solution exacte en ne tenant pas compte des erreurs d'arrondi). Pour de grands systèmes, ces méthodes demandent souvent trop d'espace mémoire et trop de calculs. On est alors amené à utiliser des méthodes itératives. Ces méthodes ne permettent pas, en général, d'obtenir la solution exacte en un nombre fini d'opérations. Mais chaque itération donne une meilleure approximation de la solution et l'on arrête dès que la précision désirée est atteinte.

2.4.1 Généralités. Définitions

Soit A carrée d'ordre n inversible , on appelle *décomposition* de A tout couple de matrices (M, N) où M est inversible, et tel que $A = M - N$.

Le système $Ax = b$ devient alors $x = M^{-1}Nx + M^{-1}b$.
En posant $B = M^{-1}N$ et $c = M^{-1}b$, la méthode itérative s'écrit

$$x^{(k+1)} = Bx^{(k)} + c \, .$$

Proposition 2.4.1 (CS)
Soit (M, N) la décomposition associée à A et supposons que pour la norme matricielle subordonnée on a $\|B\| < 1$ où $B = M^{-1}N$ Alors pour tout vecteur $x^{(0)} \in \mathbb{R}^n$, la suite $x^{(k+1)} = Bx^{(k)} + c$ est convergente.

Proposition 2.4.2 (CNS)
La suite $(x^{(k)})$ définie par $x^{(k+1)} = Bx^{(k)} + c$ est convergente pour tout $x^{(0)} \in \mathbb{R}^n$, si et seulement si $\rho(B) < 1$.

On appelle *vitesse de convergence asymptotique* de la méthode le nombre

$$R = -\log_{10}(\rho(B)) = -\log_{10}(\rho(M^{-1}N))$$

Ce nombre donne une estimation du nombre de décimales correctes gagnées en passant de $x^{(k)}$ à $x^{(k+1)}$.

Afin d'obtenir des résultats précis rapidement, on veut donc que la décomposition $A = M - N$ soit telle que M soit facilement inversible et que $\rho(M^{-1}N)$ soit petit.

Définition 2.4.1
La matrice A est à diagonale strictement dominante si

$$\text{pour tout } 1 \leq i \leq n \ : |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \, .$$

2.4.2 Méthode de Jacobi

On décompose A en $A = M - N$ où $M = D$ est la diagonale de A et la matrice $-N = -K$ contient les éléments hors diagonale. On note $J = D^{-1}K$ *la matrice de* Jacobi.
La suite est alors définie par

$$x^{(0)} \text{ donné, pour } k \geq 0 : x^{(\mathrm{k}+1)} = D^{-1}Kx^{(\mathrm{k})} + D^{-1}b \, .$$

Algorithme 2.6 (Jacobi)
Après permutations éventuelles pour rendre D inversible,
une itération de la méthode de Jacobi *s'écrit :*

pour $i = 1$ ***à*** n

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right)$$

Proposition 2.4.3
Si la matrice A est à diagonale strictement dominante, alors la méthode de Jacobi *converge.*

Application : On peut implémenter directement la résolution de l'équation (2.1), page 11, sans se préoccuper de numéroter les inconnues. On vérifie que la méthode de Jacobi converge dans ce cas et on a

$$v_{m,n}^{(k+1)} = \frac{1}{4} \left(v_{m+1,n}^{(k)} + v_{m-1,n}^{(k)} + v_{m,n+1}^{(k)} + v_{m,n-1}^{(k)} + h^2 f_{m,n} \right) \, , \text{ pour } 1 \leq m, n \leq N$$

En `Octave`, il suffit de deux matrices `v` et `v1` de tailles $N+2$ dont les premières et dernières lignes et colonnes sont nulles. Une itération peut alors s'écrire

```
v1(2:end-1,2:end-1) = ( v(3:end  ,2:end-1) + v(1:end-2,2:end-1)...
          + v(2:end-1,3:end  ) + v(2:end-1,1:end-2) + h^2*f(2:end-1,2:end-1) )/4
```

2.4.3 Méthode de Gauss-Seidel

On écrit $A = M - N$ avec $M = D - E$ et $N = F$ où D est la diagonale de A et $-E$, resp. $-F$, la matrice triangulaire strictement inférieure, resp. supérieure, de A, $K = E + F$.
On note $\mathcal{L} = (D - E)^{-1}F$ *la matrice de* Gauss-Seidel.
La suite est définie par

$$x^{(0)} \text{ donné, pour } k \geq 0 : x^{(\mathrm{k}+1)} = (D - E)^{-1}Fx^{(\mathrm{k})} + (D - E)^{-1}b \, .$$

Algorithme 2.7 (Gauss-Seidel)
Après permutations éventuelles, une itération de la méthode de Gauss-Seidel *s'écrit :*

pour $i = 1$ ***à*** n

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

Proposition 2.4.4
Si la matrice A est à diagonale strictement dominante, alors la méthode de Gauss-Seidel *converge.*
On a de plus $\|\mathcal{L}\|_\infty \leq \|J\|_\infty < 1$.

2.4.4 Méthode de relaxation

Avec les notations précédentes, la décomposition de A s'écrit

$$A = M - N \text{ avec } M = \frac{1}{\omega} D - E \text{ et } N = (\frac{1}{\omega} - 1)D + F \text{ pour } \omega \in \mathbb{R}^* .$$

On note $\mathcal{L}_{rel(\omega)} = (\frac{1}{\omega} D - E)^{-1}((\frac{1}{\omega} - 1)D + F) = (D - \omega E)^{-1}((1 - \omega)D + \omega F)$.

Algorithme 2.8
Une itération de la méthode de relaxation de paramètre ω s'écrit :

pour $i = 1$ ***à*** n

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

Note : pour $\omega = 1$ on retrouve la méthode de Gauss-Seidel.

Théorème 2.4.1
On a $\rho(\mathcal{L}_{rel(\omega)}) \geq |\omega - 1|$, *une condition nécessaire de convergence de la méthode de relaxation est donc que* $0 < \omega < 2$.

Théorème 2.4.2
Soit A une matrice symétrique définie positive, alors $\rho(\mathcal{L}_{rel(\omega)}) < 1$ *pour* $0 < \omega < 2$.
Une condition nécessaire et suffisante de convergence de la méthode de relaxation est alors que $0 < \omega < 2$.

Complexité : Pour les trois méthodes présentées, le coût d'une itération est $O(n^2)$. Tant que le nombre d'itérations nécessaires pour obtenir une solution, avec la précision souhaitée, n'est pas de l'ordre de n, les méthodes itératives sont plus rapides que les méthodes directes présentées dans la section précédente.

2.5 Détermination des valeurs propres d'une matrice

Dans beaucoup d'applications, basées sur des modèles linéaires, on a besoin de connaître les valeurs propres et vecteurs propres d'une matrice. On peut citer l'étude de la stabilité de systèmes dynamiques utilisés en mécanique, économie,...
Dans la suite, on ne s'intéressera qu'à des matrices carrés réelles.

Déterminer les valeurs propres d'une matrice peut se ramener à la résolution d'une équation polynomiale.
Par exemple, la matrice $A = diag(1, 2, \ldots, 20)$ a comme polynôme caractéristique $w(\lambda) = \prod_{i=1}^{20}(\lambda - i)$, étudié page 4. On a vu qu'une petite perturbation d'un coefficient du polynôme changeait complètement la nature des racines, et donc la nature des valeurs propres de A.
Déterminer le polynôme caractéristique et calculer ses racines est une méthode numériquement instable et donc inadaptée pour obtenir les valeurs propres d'une matrice.

Par ailleurs, on sait qu'il n'y a pas de formule générale permettant d'exprimer, grâce à des opérations élémentaires, les racines d'un polynôme de degré supérieur ou égal à cinq (résultats de É. Galois et N. Abel sur la résolution par radicaux d'une équation polynomiale). La détermination des valeurs propres d'une matrice aura donc nécessairement recours à des méthodes itératives.
Comme pour la résolution d'un système linéaire, le fait que la matrice soit symétrique, tridiagonale,... permet souvent de trouver des méthodes spécifiques adaptées. Avant de rappeler quelques définitions et résultats, on va proposer un exemple où un vecteur propre d'une très grande matrice joue un rôle important.

2.5.1 La matrice Google©

Dans cette section, on va présenter une approche simple de l'algorithme *PageRank*©. Le moteur de recherche Google© utilise cet algorithme pour obtenir un classement des pages Web en fonction de leur importance. L'algorithme utilisé actuellement par Google comporte beaucoup d'autres aspects qui ne seront pas discutés ici.

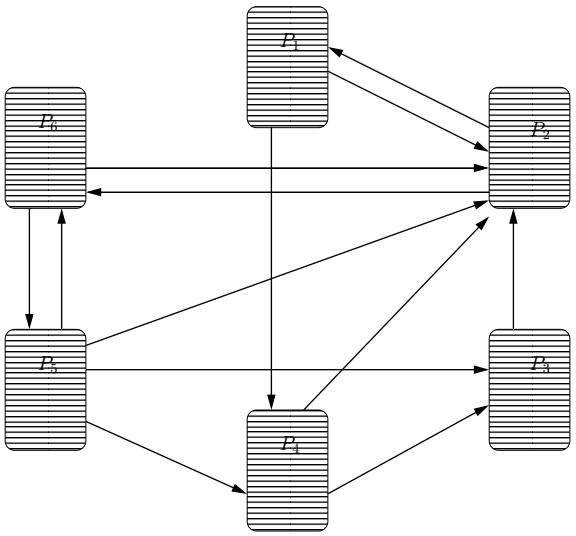
On considère n pages Web P_i, $1 \leq i \leq n$, dont on désire donner un classement par importance (ou popularité). Notons x_i l'importance de la page P_i. On va définir l'importance de la page P_i en tenant compte de l'importance des pages qui pointent vers P_i tout en pénalisant celles qui proposent trop de références vers d'autres pages.
On note $L_i \subset \{1, \ldots, n\} \setminus \{i\}$ l'ensemble des pages qui ont une référence vers P_i et s_i le nombre de pages qui sont référencées par la page P_i. En termes de graphes (voir la figure), $|L_i|$ compte le nombre d'arcs entrants et s_i le nombre d'arcs sortants de P_i.
On obtient la relation linéaire suivante

$$x_i = \sum_{j \in L_i} \frac{x_j}{s_j}, 1 \leq i \leq n\,.$$

Si on note A la matrice dont les éléments sont $a_{ij} = \begin{cases} \frac{1}{s_j} & \text{si } j \in L_i \\ 0 & \text{si } j \notin L_i \end{cases} \quad 1 \leq i, j \leq n,$

alors le vecteur $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ est solution de $Ax = x$.

Ce problème admet une solution si $\lambda = 1$ est une valeur propre de A et x un vecteur propre associé.



Exemple d'un Web à $n = 6$ pages, les flèches indiquent les liens vers d'autres pages.

Pour illustrer, on propose un exemple de référencement pour un Web de 6 pages. Par exemple $L_1 = \{2\}$, $s_1 = 2$ et $L_2 = \{1, 3, 4, 5, 6\}$, $s_2 = 2$ *etc.* On obtient ainsi la matrice

$$A = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{4} & 0 \end{pmatrix}$$

En déterminant le spectre de A, on vérifie que $\lambda = 1$ est valeur propre

$$\text{spec}(A) = \{-0.7854364\,, -0.3385082\,, 0.2184157\,, 1\,, -0.0472355 \pm i0.5165588\}$$

et un vecteur propre associé à la valeur propre 1 est de la forme $x = c \begin{pmatrix} 28 \\ 56 \\ 13 \\ 18 \\ 16 \\ 32 \end{pmatrix}$, $c \in \mathbb{R}^*$.

Prenons $c > 0$ tel que $\sum_{i=1}^{n} x_i = 1$, d'où $x = \begin{pmatrix} 0.1717791 \\ 0.3435583 \\ 0.0797546 \\ 0.1104294 \\ 0.0981595 \\ 0.1963190 \end{pmatrix}$.

De ce vecteur, on déduit l'ordre d'importance décroissante suivant : $P_2, P_6, P_1, P_4, P_5, P_3$.

On peut noter qu'un changement de numérotation des pages revient à remplacer A par une matrice semblable et ne changera pas les importances. Seuls les coordonnées de x seront permutées.

Dans cet exemple la valeur propre 1 est simple et c'est la plus grande valeur propre en module. La matrice A est stochastique par colonnes, c'est-à-dire que tout les coefficients sont positifs et que la somme par colonnes vaut 1.

Si certaines pages ne renvoient vers aucune autre, la matrice A aura des colonnes nulles. On remplace alors les colonnes identiquement nulles par le vecteur colonne $v = (\frac{1}{n} \cdots \frac{1}{n})^t \in \mathcal{M}(n,1)$ afin d'obtenir une matrice stochastique par colonnes $S = A + d * v^t$
où le vecteur $d \in \mathbb{R}^n$ est défini par $d_i = 1$ si la i^{e} page n'envoie aucun lien, 0 sinon.

Si les pages sont regroupés par îlots sans liens, on a un graphe non connexe et l'on ne pourra pas classer les pages globalement. Afin de remédier à ceci, on introduit un facteur d'amortissement $\alpha \in]0,1[$ qui permet d'arriver sur n'importe quelle page du Web. Pour $\alpha = 0.85$, on obtient la matrice Google introduite par les fondateurs de Google [1,2]

$$G = \alpha(A + d * v^t) + (1 - \alpha)\,\text{ones(n, 1)} * v^t \ .$$

Cette construction de la matrice G permet d'assurer que la valeur propre 1 est simple et la plus grande en module. Afin de déterminer la solution de $Gx = x$ pour n égal à quelques milliards, on utilise alors la méthode de la puissance adaptée aux grandes matrices, stochastiques par colonne, creuses.

Dans notre exemple, un calcul direct donne

$$A^{70} = \begin{pmatrix} 0.1717791 & 0.1717791 & 0.1717791 & 0.1717791 & 0.1717791 & 0.1717791 \\ 0.3435583 & 0.3435583 & 0.3435583 & 0.3435583 & 0.3435583 & 0.3435583 \\ 0.0797546 & 0.0797546 & 0.0797546 & 0.0797546 & 0.0797546 & 0.0797546 \\ 0.1104294 & 0.1104295 & 0.1104294 & 0.1104294 & 0.1104294 & 0.1104294 \\ 0.0981595 & 0.0981595 & 0.0981595 & 0.0981595 & 0.0981595 & 0.0981595 \\ 0.1963190 & 0.1963190 & 0.1963190 & 0.1963190 & 0.1963190 & 0.1963190 \end{pmatrix}$$

Mais si on applique la méthode de la puissance inverse (algorithme 2.10), avec $\mu = 1.1$,

et si on part du vecteur $x^{(0)} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$, on obtient $x^{(7)} = \begin{pmatrix} 0.1717791 \\ 0.3435583 \\ 0.0797546 \\ 0.1104294 \\ 0.0981595 \\ 0.1963190 \end{pmatrix}$.

1. S. Brin et L. Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Stanford University 1998

2. K. Bryan et T. Leise, *The $25,000,000,000 eigenvector : the linear algebra behind Google*, SIAM Review, 2006

2.5.2 Définitions. Stabilité des valeurs propres. Conditionnement

Dans toute cette section A sera une matrice carrée d'ordre $n \in \mathbb{N}^*$, réelle.

Définition 2.5.1

1. *On appelle valeur propre de A, toute racine $\lambda_i \in \mathbb{C}$ du polynôme caractéristique $p(\lambda) = \det(A - \lambda I_n)$. La multiplicité algébrique $m_i \in \mathbb{N}^*$ est l'ordre de multiplicité de λ_i en tant que racine de p. Si on note $r \leq n$ le nombre de racines distinctes de p, on a*
$$p(\lambda) = \prod_{i=1}^{r} (\lambda - \lambda_i)^{m_i}$$
2. *Un vecteur propre $v_i \in \mathbb{C}^n \setminus \{0\}$, associé à la valeur propre λ_i, vérifie*
$$Av_i = \lambda_i v_i.$$
On a donc $v_i \in \ker(A - \lambda_i I_n) \setminus \{0\}$ et on appelle multiplicité géométrique de λ_i l'entier naturel $g_i = \dim(\ker(A - \lambda_i I_n))$, $1 \leq i \leq r$.

La proposition suivante rappelle quelques résultats reliants la multiplicité des valeurs propres à la réduction des matrices.

Proposition 2.5.1
Soit $A \in \mathcal{M}(n, n)$, on a les propriétés :

1. *Les multiplicités vérifient les relations :*
$$\text{pour } 1 \leq i \leq r : \ g_i \leq m_i \,, \qquad \sum_{i=1}^{r} m_i = n \quad \text{et} \quad \sum_{i=1}^{r} g_i \leq n \,.$$
2. *La matrice A est diagonalisable si et seulement si $\sum_{i=1}^{r} g_i = n$.*
3. *Si toutes les valeurs propres $\lambda_i \in \mathbb{C}$, $1 \leq i \leq n$, sont simples (*i.e. *$m_i = g_i = 1$ et $r = n$), alors la matrice est diagonalisable (dans $\mathbb{C}$).*
4. *La matrice A n'est pas diagonalisable si et seulement si il existe au moins une valeur propre λ_i telle que $m_i > g_i \geq 1$.*
Dans ce cas la matrice, respectivement la valeur propre, est dite défective.

La proposition suivante permet de ramener la détermination des valeurs propres de A à la détermination des valeurs propres d'une matrice plus "simple". Il suffit (!) de multiplier A par des matrices adaptées.

Proposition 2.5.2
Soit P une matrice carrée d'ordre n, inversible. On dit que $B = P^{-1}AP$ et A sont des matrices semblables.
Alors A et B ont les mêmes valeurs propres et v est un vecteur de propre de A si et seulement si $P^{-1}v$ est un vecteur propre de B.

Afin de traiter tous les cas, diagonalisable ou défective, pour une matrice réelle A, on a recours à la décomposition de Schur réelle, voir aussi page 16.

Proposition 2.5.3 (Décomposition de Schur réelle)
Soit A une matrice carrée réelle, alors il existe une matrice réelle et orthogonale U et une matrice triangulaire supérieure par blocs T tels que que $U^tAU = T$.
Sur la diagonale de T se trouvent des blocs 1×1 correspondant aux valeurs propres réelles de A et des blocs 2×2 correspondant aux valeurs propres complexes conjuguées. de A.

Exemple :

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = UTU^t.$$

Ici $\mathrm{spec}(A) = \{\lambda_1 = 1, \lambda_2 = 1+i, \lambda_3 = 1-i\}$, donc $r = 3$ et $m_1 = 2 > g_1 = 1$, $m_2 = m_3 = g_2 = g_3 = 1$. La matrice A est défective car λ_1 est une valeur propre défective.

Nous allons nous intéresser maintenant à la stabilité du problème de détermination des valeurs propres d'une matrice. On remarque que la notion de conditionnement est différente que celle rencontrée lors de la résolution des systèmes linéaires.

Théorème 2.5.1 (Bauer-Fike, 1960)
On suppose que A est diagonalisable et que P est telle que $P^{-1}AP = D = diag(\lambda_1, \ldots, \lambda_n)$. Soit $\delta A \in \mathcal{M}(n,n)$ et $\mu \in \mathrm{spec}(A+\delta A)$, alors

$$\min_{1\le k\le n} |\mu - \lambda_k| \le c_p(P)\|\delta A\|_p\,, \qquad \textit{pour } p \in \{1, 2, \infty\}\,.$$

Ce résultat montre que le conditionnement du problème des valeurs propres de A ne dépend pas du conditionnement de A mais du conditionnement des matrices de passage P.

Si la matrice A est symétrique, elle est diagonalisable grâce à une matrice orthogonale U et $U^tAU = D$ avec $c_2(U) = 1$, donc pour $\mu \in \mathrm{spec}(A+\delta A)$

$$\min_{1\le k\le n} |\mu - \lambda_k| \le \|\delta A\|_2$$

L'exemple classique suivant présente un cas où le problème des valeurs propres est mal conditionné.

Exemple : Soit $\varepsilon \in \mathbb{R}_+$, on considère la matrice carrée d'ordre $n \ge 2$ donnée par

$$A(\varepsilon) = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ 0 & 0 & & \cdots & 0 & 1 \\ \varepsilon & 0 & & \cdots & 0 & 0 \end{pmatrix}$$

La matrice $A(0)$ n'est pas diagonalisable, elle admet $\lambda_1 = 0$ comme valeur propre de multiplicité algébrique n et de multiplicité géométrique 1.

Par contre pour $\varepsilon > 0$, la matrice $A(\varepsilon)$ est diagonalisable (sur $\mathbb{C}$) car elle a n valeurs propres distinctes $\mu_k = \sqrt[n]{\varepsilon} e^{i\frac{2(k-1)\pi}{n}}$, $1 \le k \le n$.

On a donc $|\lambda_1 - \mu_k| = \sqrt[n]{\varepsilon}$, $1 \le k \le n$. Si $\varepsilon = 10^{-16}$ et $n = 16$, on aura $|\lambda_1 - \mu_k| = 10^{-1}$, une perturbation d'ordre 10^{-16} sur un coefficient de la matrice est répercutée sur les valeurs propres avec une variation du module de 10^{-1} !

2.5.3 Méthode de la puissance

Une méthode des plus simples pour calculer les valeurs propres d'une matrice A est basée sur la méthode de la puissance. Cette méthode permet de déterminer la valeur propre de plus grand module et un vecteur propre associé. On obtient donc le rayon spectral d'une matrice

ALGORITHME 2.9
Détermination de valeur et vecteur propre par la méthode de la puissance

$x^{(0)} \in \mathbb{R}^n$ *donné,* $k = 0$
répéter
$y^{(k+1)} = Ax^{(k)}$
$x^{(k+1)} = \dfrac{y^{(k+1)}}{\|y^{(k+1)}\|}$
$\tilde{\lambda}^{(k+1)} = (x^{(k)})^t \, Ax^{(k)}$
$k = k + 1$
jusqu'à *[test d'arrêt]*

Le test d'arrêt est souvent basé sur $|\tilde{\lambda}^{(k+1)} - \tilde{\lambda}^{(k)}|$ et peut comporter un nombre d'itérations maximal afin d'éviter une boucle qui ne se termine jamais. Un résultat de convergence pour l'algorithme précédent est fourni par le théorème suivant.

THÉORÈME 2.5.2
Soit A une matrice réelle diagonalisable de valeurs propres $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$ et $\{v_1, \ldots, v_n\}$ une base de vecteurs propres normalisés, $\|v_i\|_2 = 1$, pour $1 \leq i \leq n$.

On suppose que $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ et $x^{(0)} = \displaystyle\sum_{i=1}^{n} \alpha_i v_i$, avec $\alpha_1 \neq 0$.

Alors la suite $(x^{(k)})_k$, définie par l'algorithme 2.9, vérifie

$$x^{(k)} = \frac{\alpha_1}{c_k} \lambda_1^k \, (v_1 + r_k), \qquad \text{avec } \|r_k\|_2 = O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \, , \; c_k \in \mathbb{R}_+^* \text{ tel que } \|x^{(k)}\|_2 = 1$$

et $\displaystyle\lim_{k \to +\infty} \tilde{\lambda}^{(k)} = \lambda_1 \, .$

Remarques :

1. D'après les hypothèses, λ_1 est une valeur propre simple réelle. La méthode de la puissance ne permet pas d'accéder à des valeurs propres complexes.
2. L'hypothèse que $\alpha_1 \neq 0$ est en général vérifiée pour $x^{(0)}$ choisi de façon aléatoire.
3. La vitesse de convergence dépend du rapport $|\lambda_2|/|\lambda_1| < 1$, si ce rapport se rapproche de 1, la convergence devient très lente.

Afin d'améliorer la vitesse de convergence de l'algorithme 2.9 ou de s'approcher d'une autre valeur propre λ_p, on modifie la méthode de la puissance.

Méthode de la puissance inverse

Soit μ une bonne approximation de la valeur propre λ_p, $1 \leq p \leq n$, on a alors

$$\frac{1}{|\mu - \lambda_p|} \gg \frac{1}{|\mu - \lambda_i|} \quad \text{pour } 1 \leq i \neq p \leq n \, .$$

On applique la méthode de la puissance à la matrice $(A-\mu I_n)^{-1}$ dont les vecteurs propres sont ceux de la matrice A, c'est-à-dire $\{v_1, \ldots, v_n\}$, et les valeurs propres sont les $(\lambda_i-\mu)^{-1}$, $1 \leq i \leq n$.

Il suffit d'adapter l'algorithme 2.9 pour obtenir :

ALGORITHME 2.10
Méthode de la puissance inverse
$x^{(0)} \in \mathbb{R}^n$ *donné,* $k = 0$
répéter
$y^{(k+1)} = (A - \mu I_n)^{-1} x^{(k)}$
$x^{(k+1)} = \dfrac{y^{(k+1)}}{\|y^{(k+1)}\|}$
$\tilde{\lambda}^{(k+1)} = (x^{(k)})^t \, A x^{(k)}$
$k = k + 1$
jusqu'à *[test d'arrêt]*

Remarques :

1. En utilisant ce qui précède on vérifie que la suite des $(x^{(k)})_k$ va tendre vers un multiple de v_p et la suite des $\tilde{\lambda}^{(k)}$ tend vers λ_p.
2. On calcule une fois pour toutes une décomposition LU de la matrice $A-\mu I_n$ afin de résoudre le système $(A - \mu I_n) y^{(k+1)} = x^{(k)}$. Le coût d'une itération de l'algorithme 2.10 est alors le même que celui d'une itération de l'algorithme 2.9.

Exemple : Soit $A = \begin{pmatrix} -3 & 0 & 0 \\ 17 & 13 & -7 \\ 16 & 14 & -8 \end{pmatrix}$, avec $\text{spec}(A) = \{\lambda_1 = 6, \lambda_2 = -3, \lambda_3 = -1\}$.

En appliquant la méthode de la puissance avec

$$x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{on obtient} \quad x^{(1)} = \begin{pmatrix} -0.0938417 \\ 0.7194529 \\ 0.6881724 \end{pmatrix}, \; x^{(10)} = \begin{pmatrix} 0.0002303 \\ 0.7070492 \\ 0.7071643 \end{pmatrix}$$

et $\tilde{\lambda}^{(1)} = 4.1986301\,,\ \tilde{\lambda}^{(10)} = 6.0036631.$

Si l'on applique la méthode de la puissance inverse, avec $\mu = 0$, et en partant de

$$x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{on obtient} \quad x^{(1)} = \begin{pmatrix} -0.1393466 \\ 0.6270597 \\ 0.7664063 \end{pmatrix}, \; x^{(10)} = \begin{pmatrix} 0.0000151 \\ -0.4472257 \\ -0.8944211 \end{pmatrix}$$

et $\tilde{\lambda}^{(1)} = 0.5242718\,,\ \tilde{\lambda}^{(10)} = -1.0000478.$

2.5.4 Algorithme QR

L'algorithme QR de détermination des valeurs propres de la matrice A est basé sur la factorisation QR de la matrice A qui sera présenté en premier.

Factorisation QR

THÉORÈME 2.5.3 (FACTORISATION QR)
Soit A une matrice réelle de taille $[n, n]$ telle que $\text{rang}(A) = n$ *. Alors il existe une unique matrice orthogonale Q de taille $[n, n]$, $Q^tQ = I_n$ et une unique matrice R de taille $[n, n]$ triangulaire supérieure dont les éléments diagonaux sont strictement positifs, $r_{ii} > 0$, telles que $A = QR$.*

La matrice $R = L^t$ où L est la matrice unique obtenue par factorisation de Cholesky de la matrice symétrique définie positive $A^tA = L\,L^t$.
La matrice $Q = [q_1 \cdots q_n]$ contient la base orthonormée obtenue à partir de la base de $\mathbb{R}^n$ formé par les vecteurs colonnes de $A = [a_1 \cdots a_n]$. On l'obtient par la méthode de Gram-Schmidt que l'on rappelle ici.

ALGORITHME 2.11
Décomposition QR par la méthode classique de Gram-Schmidt

pour $i = 1$ ***à*** n
  $q_i = a_i$
  pour $j = 1$ ***à*** $i - 1$
    $r_{ji} = q_j^t\, a_i$ *// coordonnée de a_i p.r. à q_j*
    $q_i = q_i - r_{ji} q_j$
  $r_{ii} = \|q_i\|_2$
  si $r_{ii} < \epsilon$ ***alors***
    Erreur : a_i est l.d. de $\{a_1, \ldots, a_{i-1}\}$
  $q_i = \dfrac{1}{r_{ii}} a_i$

Cet algorithme est très instable numériquement et n'est pas utilisé en pratique. On verra plus loin une autre méthode pour obtenir les matrices Q et R.

Itération QR

L'algorithme QR a été proposé de façon indépendante en 1961 par Vera N. Kublanovskaya et John G.F. Francis. La version simple de l'algorithme se présente comme suit.

ALGORITHME 2.12
Algorithme QR simple

$A_0 = A$, $k = 0$
répéter
  Factorisation $Q_i R_i = A_i$
  $A_{i+1} = R_i Q_i$
  $k = k + 1$
jusqu'à *[test d'arrêt]*

Le résultat suivant fournit une condition de convergence.

THÉORÈME 2.5.4
Si toutes les valeurs propres de A sont de modules distincts : $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, alors l'algorithme QR converge vers une matrice triangulaire supérieure contenant les valeurs propres de A sur la diagonale.

Tel que présenté, l'algorithme QR (algorithme 2.12) n'est pas utilisable en pratique. Un certains nombre de modifications sont à apporter :

1. Donner un algorithme de factorisation QR numériquement stable. Pour cela l'on peut utiliser les transformations de Householder présentées plus loin.
2. Diminuer la complexité de l'itération QR et accélérer la vitesse de convergence.
3. Adapter l'algorithme de façon à converger vers une matrice triangulaire supérieure par blocs (décomposition de Schur réelle) et ainsi obtenir les valeurs propres complexes conjuguées de la matrice réelle A.

Matrices de Householder

DÉFINITION 2.5.2
On appelle matrice de Householder toute matrice de la forme $H = I_n - 2u\,u^t$ où $u \in \mathbb{R}^n$ et $\|u\|_2 = 1$.

On vérifie que H est une matrice symétrique, $H^t = H$, et orthogonale, $H\,H^t = I_n$, c'est donc une involution. Une matrice de Householder correspond à une symétrie orthogonale par rapport à l'hyperplan perpendiculaire à u, c'est-à-dire une réflexion.

Soit $x \in \mathbb{R}^n$ tel que $\alpha = (x_k^2 + \ldots + x_n^2)^{1/2} > 0$ $(1 \le k \le n)$ et soit $v \in \mathbb{R}^n$ tel que

$$v_i = \begin{cases} 0 & \text{pour } 1 \le i \le k-1 \\ x_k - \alpha & \text{pour } i = k \\ x_i & \text{pour } k+1 \le i \le n \end{cases}$$

On a $\|v\|_2^2 = 2\alpha(\alpha - x_k)$ et $2v^t x/\|v\|_2^2 = 1$. Si on pose $H = I - \dfrac{2}{\|v\|_2^2} v\,v^t$, on a

$$Hx = x - v \quad \text{donc} \quad (Hx)_i = \begin{cases} x_i & \text{pour } 1 \le i \le k-1 \\ \alpha & \text{pour } i = k \\ 0 & \text{pour } k+1 \le i \le n \end{cases}$$

Les $k-1$ premières cordonnées de x restent inchangées, les $n-k$ dernières coordonnées sont annulées et la k-ième de vient $(Hx)_k = \alpha > 0$.

Pour obtenir une factorisation QR de la matrice A qui soit stable numériquement, on peut utiliser les matrices de Householder.

Notons $A = [x_1 \cdots x_n]$ où x_k, $1 \le k \le n$, est la k-ième colonne de A et posons $A_0 = A$.

Pour $1 \le k \le n$ on calcule $A_k = H_k A_{k-1}$ où la matrice H_k est construite de façon à annuler les $n-k$ dernières coordonnées de la k-ième colonne de A_{k-1}.
On vérifie que $H_k A_{k-1}$ laisse invariant les $k-1$ premières lignes et colonnes de A_{k-1}.

On obtient donc $H_n \cdots H_1 A = A_n$ où $A_n = R$ est triangulaire supérieure avec des éléments strictement positifs sur la diagonale.

Donc, $A = (H_1 \cdots H_n) A_n = QR$ où $Q = H_1 \cdots H_n$ vérifie $Q^t Q = I_n$.

En pratique, on ne construit pas la matrice H_k pour calculer $H_k A_{k-1}$.
La fonction `Octave qr_housholder()`, proposé ci-dessous, utilise uniquement les coordonnées de k à n pour v et les blocs `(k:n,k:n)` pour R.

```
1  % Decomposition QR par matrices de Housholder
2  function [Q,R] = qr_housholder(A)
3   R = A
4   Q = eye(A)
5   n = size(A,1)
6   for k=1:n-1
7      v = R(k:n, k)
8      a = norm(v)
9      if (v(1)<0.0)
10        v(1) = v(1) - a
11     else
12        v(1) = - (v(2:end)'*v(2:end)) / ( v(1)+a )
13     end
14     b = 2/(v'*v)
15     R(k:n,k:n) = R(k:n,k:n) -b*v*(v'*R(k:n,k:n))
16     Q(: , k:n) = Q(:,k:n) -b*(Q(:,k:n)*v)*v'
17  end
18  if R(n,n)<0 then
19     R(n,n) = abs( R(n,n) )
20     Q(:,n) = -Q(:,n)
21  end
22 endfunction
```

Exemple :

Soit $A = \begin{pmatrix} 4 & 4 & 4 & 1 \\ 3 & 5 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 5 & 5 \end{pmatrix}$ avec $\text{spec}(A) = \{10.29516, 4.86478, 0.73567, -0.89563\}$.

On détermine la décomposition QR de A en utilisant les matrices de Householder, on obtient successivement :

$$H_1 = \begin{pmatrix} 0.7698 & 0.5773 & 0.1924 & 0.1924 \\ 0.5773 & -0.4480 & -0.4826 & -0.4826 \\ 0.1924 & -0.4826 & 0.8391 & -0.1608 \\ 0.1924 & -0.4826 & -0.1608 & 0.8391 \end{pmatrix} \quad A_1 = H_1 A_0 = \begin{pmatrix} 5.1961 & 6.5433 & 5.3886 & 2.6943 \\ 0 & -1.3787 & -1.4826 & -3.2493 \\ 0 & -0.1262 & -0.1608 & 0.5835 \\ 0 & -1.1262 & 3.8391 & 3.5835 \end{pmatrix}$$

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.7725 & -0.0707 & -0.6310 \\ 0 & -0.0707 & 0.9971 & -0.0251 \\ 0 & -0.6310 & -0.0251 & 0.7753 \end{pmatrix} \quad A_2 = H_2 A_1 = \begin{pmatrix} 5.1961 & 6.5433 & 5.3886 & 2.6943 \\ 0 & 1.7847 & -1.2658 & 0.2075 \\ 0 & 0 & -0.1522 & 0.7214 \\ 0 & 0 & 3.9162 & 4.8142 \end{pmatrix}$$

$$H_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.0388 & 0.9992 \\ 0 & 0 & 0.9992 & 0.0388 \end{pmatrix} \quad A_3 = H_3 A_2 = \begin{pmatrix} 5.1961 & 6.5433 & 5.3886 & 2.6943 \\ 0 & 1.7847 & -1.2658 & 0.2075 \\ 0 & 0 & 3.9192 & 4.7826 \\ 0 & 0 & 0 & 0.9079 \end{pmatrix}$$

Comme $H_4 = I_4$, on a $R = A_3$ et $Q = H_1 H_2 H_3$ d'où

$$A = QR = \begin{pmatrix} 0.7698 & -0.5810 & -0.2254 & 0.1375 \\ 0.5773 & 0.6848 & -0.0623 & -0.4402 \\ 0.1924 & 0.4150 & 0.1246088 & 0.8804 \\ 0.1924 & -0.1452 & 0.9642 & -0.1100 \end{pmatrix} \begin{pmatrix} 5.1961 & 6.5433 & 5.3886 & 2.6943 \\ 0 & 1.7847 & -1.2658 & 0.2075 \\ 0 & 0 & 3.9192 & 4.7826 \\ 0 & 0 & 0 & 0.9079 \end{pmatrix}$$

En appliquant l'algorithme QR 2.12, on obtient après 20 itérations

$$T_{20} = \begin{pmatrix} 10.2952 & -0.06007 & 3.7399 & 0.31193 \\ 1.3 \cdot 10^{-06} & 4.86479 & 1.20105 & 0.92827 \\ 0 & 0 & -0.97705 & -1.52341 \\ 0 & 0 & 0.09153 & 0.81709 \end{pmatrix}$$

Il faut 40 itérations pour obtenir une approximation de toutes les valeurs propres de A :

$$T_{40} = \begin{pmatrix} 10.2952 & -0.06007 & -3.7149 & 0.53260 \\ 0 & 4.86479 & -1.14404 & 0.99769 \\ 0 & 0 & -0.88622 & 1.62434 \\ 0 & 0 & 0.00940 & 0.72626 \end{pmatrix}$$

On améliore la vitesse de convergence en utilisant une technique de translation (angl. *shift*).

2.5.5 Méthode de Jacobi pour matrices réelles symétriques

Soit $A = (a_{ij})_{1 \le i,j \le n}$ une matrice réelle symétrique. La méthode de Jacobi (1846) construit une suite de matrices $A = A_0, A_1, \ldots, A_k, \ldots$, qui converge vers une matrice diagonale D.
On a $A_{k+1} = J_k^t \, A_k \, J_k$, où J_k est une rotation de Jacobi, $J_k^t J_k = I_n$.
Si l'on pose $J = J_0 J_1 \cdots J_{k-1}$, on aura $D \approx J^t \, A \, J$, resp. $A \approx J \, D \, J^t$. Comme J est encore une matrice orthogonale, D contient une approximation de toutes les valeurs propres de A et les colonnes de J forment une base orthonormée de vecteurs propres.

Une rotation de Jacobi est définie grâce à une matrice de Givens :

$$G(i,j,\theta) = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccccc} & & & & & i & & j & & \\ \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & & 0 \\ 0 & 1 & & & & & & & \vdots \\ \vdots & & \ddots & & & & & & \\ 0 & & & c & & -s & & & 0 \\ \vdots & & & & \ddots & & & & \vdots \\ 0 & & & s & & c & & & 0 \\ \vdots & & & & & & \ddots & & \vdots \\ & & & & & & & 1 & 0 \\ 0 & \cdots & & 0 & \cdots & 0 & \cdots & 0 & 1 \end{pmatrix} \end{array} \begin{array}{l} \\ \\ \\ i \\ \\ j \\ \\ \\ \end{array}$$

avec $c = \cos\theta$ et $s = \sin\theta$.
Pour $x \in \mathbb{R}^n$, $G(i,j,\theta)x$ représente une rotation d'angle θ dans le plan $\Pi(e_i, e_j)$. De même le produit $G(i,j,\theta)^t A$ n'affecte que les i^{e} et j^{e} lignes de A, tandis que $AG(i,j,\theta)$ ne transforme que les i^{e} et j^{e} colonnes de A.
En particulier, le coût de la multiplication de A par $G(i,j,\theta)$ est en $O(n)$.

Les matrices de Givens sont souvent utilisées pour annuler, par multiplication, un élément donné d'une matrice A.
Dans la méthode de Jacobi, la matrice de Givens est construite grâce à une valeur de θ qui permet d'annuler les éléments a_{ij} et a_{ji}. On montre que θ vérifie alors

$$\operatorname{cotg}(2\theta) = \frac{a_{ii} - a_{jj}}{2a_{ij}} = \tau$$

De cette relation on peut déduire les valeurs de c et s grâce à l'équation vérifiée par $t = \frac{s}{c} = \operatorname{tg}\theta$, $t^2 + 2\tau t - 1 = 0$. L'algorithme 2.13 remplace la matrice A par une matrice $\tilde{A}$ telle que $\tilde{a}_{ij} = \tilde{a}_{ji} = 0$.

Algorithme 2.13
Rotation de Jacobi pour annuler les éléments de coordonnées $\{i, j\}$

si $|a_{ij}| > \epsilon$ ***alors***

$$\tau = \frac{a_{ii} - a_{jj}}{2a_{ij}}$$

$$t = \frac{\text{signe}(\tau)}{|\tau| + \sqrt{1 + \tau^2}}$$

$$c = \frac{1}{\sqrt{1 + t^2}} \quad \textit{et} \quad s = ct$$

$$\tilde{A} = G(i,j,\theta)^t \, A \, G(i,j,\theta)$$

Si on note $S(A) = \sum_{1 \le p < q \le n} a_{pq}^2$, alors $S(\tilde{A}) = S(A) - a_{ij}^2$. On a ainsi un moyen de mesurer la convergence vers une matrice dont les éléments hors diagonale sont petits. D'où l'algorithme 2.14 qui converge vers une matrice diagonale contenant les valeurs propres, et une matrice J contenant les vecteurs propres de A.

ALGORITHME 2.14
Algorithme de Jacobi pour valeurs et vecteurs propres d'une matrice symétrique

Seuil $\sigma > 0$ *donné,* $J = I_n$, $k = 0$
répéter
Choisir $\{i,j\}$ *et en déduire* $J_k = G(i,j,\theta)$ *par l'algorithme 2.13*
$A = J_k^t \, A \, J_k$
$J = J \, J_k$
$k = k + 1$
jusqu'à $\max_{1 \le p < q \le n} |a_{pq}| < \sigma$

Dans l'algorithme original de Jacobi, on choisit les coordonnées $\{i,j\}$ de façon à avoir $|a_{pq}| = \max_{1 \le p < q \le n} |a_{kl}|$. Ceci permet d'avoir une bonne vitesse de convergence. Par contre, pour n grand, la recherche du maximum parmi les $(n^2 - n)/2$ éléments hors diagonale va prendre plus de temps par itération que la multiplication par la matrice de Givens. En pratique on parcourt A tout simplement ligne par ligne par exemple.

Exemple : Soit $A = \begin{pmatrix} 2 & 1 & -3 \\ 1 & -6 & 3 \\ -3 & 3 & 0 \end{pmatrix}$ et l'on fixe $\sigma = 10^{-5}$.

On a $\mathrm{spec}(A) = \{-7.7155084\,, -0.5472851\,, 4.2627936\}$ et $S(A) = 19$.

Pour $J_0 = \begin{pmatrix} 0.8112422 & 0 & 0.5847103 \\ 0 & 1 & 0 \\ -0.5847103 & 0 & 0.8112422 \end{pmatrix}$, on annule $a_{13} = a_{31} = -3$:

$$A_1 = J_0^t \, A \, J_0 = \begin{pmatrix} 4.1622777 & -0.9428887 & 2.22\,10^{-16} \\ -0.9428887 & -6 & 3.0184368 \\ 2.22\,10^{-16} & 3.0184368 & -2.1622777 \end{pmatrix} \text{ avec } S(A_1) = 10.$$

Après 5 itérations, on a $J_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.9999904 & -0.0043757 \\ 0 & 0.0043757 & 0.9999904 \end{pmatrix}$ et

$$A_5 = J_4^t \, A_4 \, J_4 = \begin{pmatrix} 4.2627928 & 0.0029793 & -0.0000130 \\ 0.0029793 & -7.7155077 & 6.94\,10^{-18} \\ -0.0000130 & 1.22\,10^{-16} & -0.5472851 \end{pmatrix} \text{ avec } S(A_5) = 0.0000089.$$

En plus $J = J_1 J_2 J_3 J_4 = \begin{pmatrix} 0.7766476 & -0.2219207 & 0.5895504 \\ -0.1060891 & 0.8764440 & 0.4696712 \\ -0.6209376 & -0.4273139 & 0.6571448 \end{pmatrix}$.

Implémentation numérique

En pratique, le produit $J_k^t\, A_k\, J_k$ est implémenté de façon à avoir effectivement un coût $O(n)$. La fonction `Octave` suivante propose un tel code.

```
1  % Calcul de B=J'*A*J en n'utilisant
2  % que les colonnes et lignes {i,j}
3  function [B] = rotation_jacobi(A,i,j)
4    % détermination de c et s
5    if abs(A(i,j))< 0.0000001
6         printf("\n  Problème de précision : élément trop petit \n")
7         return
8    end
9    tau = (A(i,i)-A(j,j))/(2*A(i,j))
10   t = sign(tau) / ( abs(tau) + sqrt(1+tau^2) )
11   c = 1/sqrt(1+t^2)
12   s = c*t
13   % Calcul de C = [ B(:,i),  B(:,j) ]
14   C = [ A(:,i) , A(:,j) ] * [ c, -s ; s, c]
15   C( [i,j] , : ) = [ c, s ; -s, c] * C( [i,j] , : )
16   % Construction de B par symétrie
17   B = A
18   B( : , [i,j] ) = C
19   B( [i,j] , : ) = C'
20 endfunction
```